

EXHIBIT 11

I R E L L & M A N E L L A L L P

A REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

1800 AVENUE OF THE STARS, SUITE 900
LOS ANGELES, CA 90067-4276
TELEPHONE (310) 277-1010
FACSIMILE (310) 203-7199

840 NEWPORT CENTER DRIVE, SUITE 400
NEWPORT BEACH, CALIFORNIA 92660-6324

TELEPHONE (949) 760-0991
FACSIMILE (949) 760-5200
WEBSITE: www.irell.com

WRITER'S DIRECT
TELEPHONE (949) 760-5204
FACSIMILE (949) 760-5200
NHEFAZI@IRELL.COM

February 15, 2012

VIA EMAIL

Spencer Hosie
Hosie Rice LLP
Transamerica Pyramid, 34th Floor
600 Montgomery Street
San Francisco, CA 94111

Re: *Implicit Networks, Inc. v. Juniper Networks, Inc.*
Case No. C 10-4234 SI (N.D. Cal.)

Dear Spencer:

I write to address a number of outstanding issues relating to discovery in this case, including issues raised in your letter of February 10, 2012 and Will Nelson's letters of February 6 and 10, 2012.

Implicit's Document Production

On February 4, 2012, I wrote Implicit a letter raising concerns regarding the adequacy of Implicit's document production in response to Juniper's First Set of Requests for Production. Specifically, I indicated that a review of that production "has turned up almost no **relevant** documents" dated before 2000, raising concerns regarding the sufficiency of the "several archived volumes of electronic data" Implicit claims to have searched. As I indicated previously, the period from 1997 to 1999 is the period in which Implicit claims to have "diligently" reduced to practice the patents-in-suit, and we found it odd that despite Implicit's professed diligence during that period and the existence of an official company website at least as early as April of 1997, that Implicit's non-generic production has produced few if any relevant documents between 1997-1999.

In response, in his letter of February 10, 2012, Mr. Nelson simply identifies nearly 350 documents "dated in the period 1997 to 1999 in [Implicit's] production". But my letter did not say there were no documents from that period, rather it raised concerns with the fact that there were "almost no **relevant** documents" from that critical period in Implicit's production. And Mr. Nelson's letter has confirmed that is true; the great majority of the documents identified in Mr. Nelson's letter are wholly irrelevant and some are even

IRELL & MANELLA LLPA REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
February 15, 2012
Page 2

offensive.¹ For example, taking just the first 30 documents identified in Mr. Nelson's letter as a sample, there is not a single one that even remotely relates to reduction to practice or any other issue of relevance to the case. This is apparent by simply looking at the subject lines of the emails cited:

Last five digits of Bates #			Title/Subject		Date
06745	Comm Stock distribution info for VC, after 5-14-01	7/19/2000	16781	Can't print from Word	2/13/1997
16782	Outlook is too fast	2/16/1997	16783	Removing recycle bin	2/16/1997
16784	How do I format an NTFS partition?	2/21/1997	16785	Images won't appear	2/21/1997
16786	Removing the recycle bin	2/21/1997	16787	Creating a new profile	2/21/1997
16788	Adding columns in Excel	2/24/1997	16789	Creating new views in Outlook	2/24/1997
16790	I only have 3 gigs of disk space	2/24/1997	16791	Remote access with Exchange Client	2/24/1997
16792	Outlook forms development	2/24/1997	16793	Dual boot Windows 95 and Windows NT	2/24/1997
16794	Downloading ActiveX Controls	2/24/1997	16795	Finding a public folder	2/25/1997
16796	Printing forms from Outlook	2/25/1997	16797	test	2/26/1997
16798	Can't print from IE 3.01	2/28/1997	16799	Can't print	3/3/1997
16800	VB Script programming	3/4/1997	16801	How do I create modules for my VBA code?	3/7/1997
17242	Magic-number	7/7/2004	17370	GOTCHAS	7/5/2001
18973	Magic-number	5/24/2004	27593	Mve	5/13/2004
27777	Magic-number	5/25/2004	2152	I got a joke for you...	11/13/1998
14224	Undeliverable	9/2/1998	14902	RE: One more thing	11/3/1998
15191	Wow. Only half a meg!	9/1/1998			

¹ See, e.g., IMPLICIT_00002152 (misogynist email to Edward Balassanian).

IRELL & MANELLA LLPA REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
February 15, 2012
Page 3

Moreover, even setting aside the question of relevance, there remain significant gaps in Implicit's responsive production in which there are no documents at all. For example, during the nine-month period between April 1997 and December 1997, your production contains just **two** emails. Given that Implicit contends to have been engaged in "diligent" reduction to practice of the claimed invention during that time period, one would expect much more. Please confirm that Implicit will supplement its production with the missing documents from this period.

Implicit's Interrogatory Responses

Interrogatory No. 2

Thank you for agreeing to check with your client for additional documents that would corroborate his alleged conception and reduction to practice of the purported invention. We would also appreciate if you could check with him (or perhaps your document production vendor) to see if you could produce legible copies of Mr. Balassanian's lab notebooks.

Interrogatory No. 3

Your letter of February 10, 2012, claims to have identified all "licensing and enforcement efforts." As it stands now, however, your response to this interrogatory fails to identify any licensing or enforcement efforts outside the litigation context. I understand this to be a representation that you are unaware of any effort or attempt by Implicit to license its patents outside the litigation context. Please correct me if I am mistaken.

With respect to "the relevant facts surrounding any investigation into alleged infringement" your letter does not (and cannot) contest that Implicit's response to this interrogatory fails to provide this information. Instead, you assert that information related to any investigation into alleged infringement "by infringers other than Juniper are neither relevant to this dispute nor discoverable." First, Implicit's failure to provide this information is not limited to alleged "infringers other than Juniper." Implicit has failed to provide this information as to Juniper. Second, obviously not all the facts surrounding investigations into alleged infringement will be privileged (e.g., communications with alleged infringers or entities Implicit approached about licensing). Third, this information may be relevant in a number of ways, for example (without limitation), infringement, validity, inequitable conduct, damages, laches, marking, and others. It may also provide valuable information regarding the existence, description, nature, custody, condition and location of documents and things, as well as the identity and location of persons who know of any discoverable matter (including third parties).

IRELL & MANELLA LLP

A REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
February 15, 2012
Page 4

Interrogatory No. 5

Implicit's response to this interrogatory points to a single white paper dated after December 29, 1999. This is clearly not an adequate response. Implicit's objection to Juniper's interrogatory as overbroad does not allow Implicit to ignore it entirely, especially as Implicit has already stated that it is unilaterally narrowing the scope of the interrogatory to "publications concerning the field of art of the patents-in-suit,"

Moreover, as I indicated in my letter of February 4, 2012, this interrogatory expressly requests that Implicit identify, among other things, all versions of www.strings.com and www.becomm.com dated prior to December 29, 1999. Implicit has repeatedly failed to provide discovery on these clearly relevant websites, and Mr. Nelson's February 6, 2012, letter fails to address this issue entirely.

Interrogatory No. 6

Thank you for clarifying the date of Implicit's Demo2000. Nevertheless, this was merely an example and Implicit's response to this interrogatory remains incomplete. For example, Implicit fails to identify for which years Implicit attended CES and Comdex. Additionally, Implicit identifies just 12 documents responsive to this request, only five of which are actually dated before December 29, 1999. Indeed, the only document identified in Implicit's response that refers to Comdex is dated after December 29, 1999. Surely this cannot constitute all "related documents (including, without limitation, emails, conference guides, schedules, proceedings, agendas, articles, publications and presentation materials)" for these trade shows.

Interrogatory No. 8

I understand Implicit to be representing that it has provided all relevant factual bases that it is aware of at this point. We accept this representation with the understanding that you will not attempt to rely on additional factual bases at trial that have not been included in your interrogatory response.

Interrogatory No. 9

Thank you for confirming that INI will supplement its response to this interrogatory.

In response to nearly every issue we raised with respect to Implicit's deficient interrogatory responses, Mr. Nelson indicates that you "will consult again with [your] client to see whether additional facts or documents can be identified". Please confirm that you will provide your supplementation no later than February 22, 2012.

IRELL & MANELLA LLPA REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
February 15, 2012
Page 5

Implicit's Amended Infringement Contentions

As a preliminary matter, we disagree with your suggestions that Juniper has somehow been less than diligent in raising issues relating to Implicit's infringement contentions, and that Juniper is only raising these issues now as "a matter of strategic expediency." That is wrong.

Implicit served its original infringement contentions on May 23, 2011. At that time, as you well know, Juniper's counsel was busily engaged in preparing for and then trying a multi-week trial in the District of Massachusetts that did not end until mid-August 2011. In fact, you agreed to special accommodations to Juniper's deadlines in this case as a result of that trial schedule—for which accommodation we expressed our appreciation. As we returned our attention to this case in the beginning of September, we began to note several deficiencies in Implicit's infringement contentions. And we soon learned that you were in agreement that the infringement contentions required revision. Specifically, on September 22, 2011, Implicit sought leave to amend its infringement contentions to, among other things, drop its quality of service allegations and "add[] detail to the other contentions", which we hoped would address the issues we had begun to note in Implicit's original contentions.

Implicit delayed in providing Juniper with its amended contentions. After a number of inquiries, we finally received your amended infringement contentions on November 15, 2011, by which time the parties were already in the midst of briefing and preparing for the Markman hearing in mid-January 2012. Nevertheless, on January 2, 2012, David McPhie wrote you to identify some initial concerns we had with Implicit's amended infringement contentions and indicated that "[o]ur review of these contentions is still ongoing – as you might imagine, we have been primarily focused on preparing for the upcoming claim construction hearing".

The claim construction hearing ended on January 19, 2012. A few days later, on January 23, 2012, you received my letter identifying the deficiencies in Implicit's infringement contentions. It then took more than two weeks for you to respond to my January 23 letter. Thus, the record demonstrates that Juniper has, under the circumstances, diligently pursued its inquiries regarding Implicit's infringement contentions, and indeed Implicit has taken a number of actions that have unduly delayed this process.

Nevertheless, the timing issues are significantly less important than the need to address and (if possible) resolve the substantive deficiencies in Implicit's infringement contentions. To begin with, my letter of January 23, 2012 identified a number of products which Implicit failed to even mention in its infringement charts (i.e., the DX, J, LN, SA and T-series products). Your letter of February 10, 2012 indicates that you will "review the list

IRELL & MANELLA LLP

A REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
February 15, 2012
Page 6

of products” and “drop any that should be dropped.” Please confirm that you will let us know which products you will be dropping no later than February 22, 2012.

Relatedly, for those products Implicit chooses not to drop, please confirm that Implicit will provide charts identifying “where each limitation of each asserted claim is found within each Accused Instrumentality,” as required by Local Rule 3-1(c). As your letter of February 10, 2012 acknowledges, JUNOS is an operating system enabled on “numerous Juniper products,” each of which constitutes a different product. The implementation of JUNOS is not uniform across all these products and the JUNOS functionality that exists in one product does not necessarily exist in others. As such, Implicit’s assumption that all products operating using the JUNOS software implement the same functionality is improper and incorrect. Implicit cannot evade its obligations under Local Rule 3-1(c) by attempting to mix and match functionality from the different products implementing JUNOS.

Also, I am not entirely clear what you mean in your letter when you state that you will “ensure that [Implicit’s] Junos-enabled product list is as complete as [Implicit] can make it” (see your letter of February 10, 2012). It is too late for Implicit to add additional products to its infringement contentions. Although we are more than happy to let you drop products, if you intend to attempt to add products, please let us know right away, as we have serious concerns about the validity of such an approach.

With respect to Implicit’s failure to provide pinpoint citations to the source code, my colleague Douglas Dixon’s letter of February 14, 2012, outlines Juniper’s position on this issue. It appears we have reached an impasse, and I do not feel it necessary to say anymore on this issue other than to note that your letter of February 10, 2012, distorts the factual record and fails to address the authority I cited in my January 23, 2012. I note that it likewise appears that the parties have reached an impasse regarding the deficiencies in Implicit’s disclosure under Patent Local Rule 3-2(a). We will begin our preparations to bring these matters to the Court’s attention.

With respect to the remaining issues in my January 23, 2012 letter, I understand from my meet and confer with your colleagues, Will Nelson and George Bishop, that Implicit agrees to amend its infringement contentions to reflect the fact that Implicit has dropped claims 26 and 45 of the ‘163 patent, and also to drop its allegations under 35 U.S.C. § 271 (f)-(g) and its allegations of willfulness. I also understand that Implicit agrees to supplement its contentions by identifying the specific claim(s) of the patents-in-suit each product practices, as required by Patent Local Rule 3-1(g). Please let me know if I am mistaken in any way. Please also confirm that we will receive this supplementation no later than February 22, 2012.

IRELL & MANELLA LLPA REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
February 15, 2012
Page 7

Juniper's Production

I turn now to Implicit's allegations regarding Juniper's responses to discovery. As an initial matter, we were surprised by your belated complaint regarding the search terms Juniper proposed in its response to Implicit's Interrogatory No. 14. As you know, Juniper served its response to Implicit's Interrogatory No. 14, along with said search terms, nearly **four months** ago, on October 20, 2011. Implicit never once complained about the use of search terms or the specific terms that were proposed, nor did it provide or suggest additional search terms. It remained silent on this issue entirely. Indeed, even now Implicit has failed to propose additional search terms it believes should have been included. This suggests to us that Implicit's complaints are merely an attempt to focus attention away from its own serious discovery issues.

We also disagree with your assertion that Juniper's production lacks internal documents concerning the design, development, and architecture of the accused products and business planning and operations surrounding the accused products. A quick search through the over one million pages of documents Juniper has produced confirms that there are numerous documents in the production of the type you are requesting, including documents that go to marketing requirements and technical operation of the Juniper products. While it is not feasible to list all of these documents (there are too many), following are a few examples:

JUNIPER01686766; JUNIPER01686800; JUNIPER01686845; JUNIPER01686862;
JUNIPER01686866; JUNIPER01686886; JUNIPER01686910; JUNIPER01686916;
JUNIPER01686922; JUNIPER01686933; JUNIPER01686966; JUNIPER01686968.

Nevertheless, as a show of good faith and in the spirit of avoiding unnecessary disputes, we will agree to conduct a further review of our production and attempt to determine whether additional documentation may be available, in which case we will supplement our production. Note the productiveness of any further investigation depends in large part on Implicit's willingness to provide us with complete infringement contentions. If we are still left to guess about what Implicit is accusing of infringement, it will remain difficult, if not impossible, to identify relevant documents beyond those that we have already produced. Thus, we believe both parties would benefit from Implicit correcting the deficiencies in its infringement contentions immediately.

IRELL & MANELLA LLP

A REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
February 15, 2012
Page 8

Juniper's Interrogatory Responses

Interrogatory No. 9

Implicit states that it does "not believe it's even remotely likely" that only six patent license agreements resulted from the "scores of patent litigation" Juniper has been involved in. (*See* Will Nelson's Letter of February 6, 2012). Please identify the "scores of patent litigation" to which you are referring, which should be a matter of public record. We do not believe they exist. Nevertheless, if you have additional information on this front, let us know and we would be willing to investigate.

Interrogatory Nos. 1, 4, 7 and 10

After conducting a reasonably diligent search, my colleague, Daniel Hipskind, informed you that we were unable to identify reports summarizing the documents identified in Juniper's responses to Implicit's Interrogatory Nos. 1, 4, 7 and 10 by product line. Nevertheless, you now "seek lead counsel's affirmance of Mr. Hipskind's assertions that no summary data is available." (*See* Will Nelson's Letter of February 6, 2012). I am unaware of any obligation of lead counsel to make such an affirmance, and do not understand why Mr. Hipskind's assertion is not sufficient. Nevertheless, if you have any authority to support your position, please let me know and we will consider it.

30(b)(6) Deposition of Implicit

Finally, we still need to schedule the 30(b)(6) deposition of Implicit, although, as we have mentioned before, we need to first resolve these other open discovery issues to facilitate a meaningful deposition.

Please note that, although we had the opportunity to substantively discuss at our last meet-and-confer many of the issues set forth above, there are only a couple on which we have reached a final impasse (*i.e.*, the source code pin-cite and Patent Local Rule 3-1(a) issues). Please let me know some times later this week you would be available to further discuss the remaining issues. As you know, our preference is always to try to avoid motion practice on discovery matters that can and should be resolved between the parties.

Sincerely,

/s/ Nima Hefazi
Nima Hefazi

NH

EXHIBIT 12

I R E L L & M A N E L L A L L P

A REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

1800 AVENUE OF THE STARS, SUITE 900
LOS ANGELES, CA 90067-4276
TELEPHONE (310) 277-1010
FACSIMILE (310) 203-7199

840 NEWPORT CENTER DRIVE, SUITE 400
NEWPORT BEACH, CALIFORNIA 92660-6324

TELEPHONE (949) 760-0991
FACSIMILE (949) 760-5200
WEBSITE: www.irell.com

WRITER'S DIRECT
TELEPHONE (949) 760-5204
FACSIMILE (949) 760-5200
NHEFAZI@IRELL.COM

March 2, 2012

VIA EMAIL

Spencer Hosie
Hosie Rice LLP
Transamerica Pyramid, 34th Floor
600 Montgomery Street
San Francisco, CA 94111

Re: *Implicit Networks, Inc. v. Juniper Networks, Inc.*
Case No. C 10-4234 SI (N.D. Cal.)

Dear Spencer:

I write to follow-up on a number of outstanding discovery issues.

Implicit's Infringement Contentions And Disclosures

As we discussed during the parties' February 28, 2012, meet-and-confer, Implicit's infringement contentions remain deficient. First, as we have stated repeatedly, Implicit has failed to comply with Local Rule 3-1(c), which requires Implicit to provide a chart identifying specifically where each limitation of each asserted claim is found **within each Accused Instrumentality.** During the meet-and-confer, Implicit agreed to let us know no later than Monday, March 5, whether it would provide legally sufficient chart. If not, the parties will be at an impasse and will need to seek the guidance of the Court.

In the meantime, you asked us to consider the holding in *Implicit Networks, Inc. v. Hewlett-Packard Company*, 2011 U.S. Dist. LEXIS 100283 (N.D. Cal. Sept. 7, 2011). We have done so and it does not support Implicit's position. In fact, the Court expressly found it would be "improper" to group together in a single chart products that "actually function in different ways." *Id.* at *7.¹ That is precisely what Implicit has done in its charts for Juniper's products. Moreover, the fact that many of the accused products happen to run some version of JUNOS (which is itself not an accused product) does not justify Implicit's

¹ The Court also found in its opinion that Implicit had failed to properly make a claim under the doctrine of equivalents under Rule 3-1(e). As Implicit has made no attempt to correct or supplement the same improper boilerplate language found in the Juniper charts, we understand that Implicit intends to make no claim under the doctrine of equivalents in this case.

IRELL & MANELLA LLP

A REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
March 2, 2012
Page 2

failure to comply with the Local Patent Rules. JUNOS is a complex operating system with different functionalities implemented for different products. *See, e.g.*, Juniper 9-20-2011 30(b)(6) Depo. Tr. 91:20-92:8; *id.* at 124:15-125:2. To use an analogy, versions of the Windows operating system can run on a phone, a laptop, a server, or even an automobile, but no one would say that these disparate products “actually function” in the same way.

By failing to comply with Rule 3-1(c), Juniper is left to “guess which versions of its products infringes the patent.” *Bender v. Freescale Semiconductor, Inc.*, 2010 U.S. Dist. LEXIS 91281; *see also Network Caching Tech., LLC v. Novell, Inc.*, 2002 U.S. Dist. LEXIS 26098 (“FRCP 11 requires that a plaintiff compare an accused product to its patents on a claim by claim, element by element basis for at least one of each defendant’s products.”). Indeed, Implicit’s charts do not even mention some of the products you claim to be accusing of infringement, such as the T-series and LN-series routers. Juniper should not have to still be guessing about Implicit’s theories of infringement at this stage of the litigation.

We understand that you have offered to attempt to remedy these deficiencies in part by dropping your claims of infringement with respect to certain products (*e.g.*, Juniper’s ScreenOS-related products and stand-alone IDP products). We appreciate and hereby accept your offer to drop from this case all products you have previously accused but omitted from your most recent set of proposed infringement contentions, which you sent this week. However, we must object to Implicit’s attempt to add additional products to its list of accused products that have not been previously identified in your infringement contentions, which only exacerbates the problem under Rule 3-1(c). For example, it appears Implicit now seeks to inject into this case at least **15 new products**—*i.e.*, MX5, MX10, MX40 Router, MX80, T4000, TX Matrix and TX Matrix Plus Routers, as well as the EX2200, EX2500, EX3200, EX3300, EX4200, EX4500, EX6200 and EX8200 Ethernet Switches. As another example, products that were previously accused only in combination with certain other components are now improperly listed separately. Your proposal would thus significantly expand the scope of accused products in this case.

Although we are not presently inclined to enter into a stipulation that would permit Implicit to amend its infringement contentions to expand its list of accused products, if you feel there is a legitimate reason that you could not have accused these products earlier, please let us know, and we would be happy to meet-and-confer with you further on this issue. If we cannot reach agreement on this point, we may need to inform the Court and seek further guidance.

We further understand that Implicit is continuing to look into our requests regarding compliance with Rule 3-2(e), and we look forward to hearing from you by Monday whether we can expect to receive the additional information we have requested.

IRELL & MANELLA LLP

A REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
March 2, 2012
Page 3

Implicit's February 22, 2012 Production

We received a production Bates stamped IMP104762 – IMP120577 on February 23, 2012. While our review is ongoing, an initial analysis of the documents has uncovered a number of serious deficiencies. We request Implicit immediately correct them.

First, throughout the production, single documents have been fragmented into hundreds of discrete records. Many of these single documents have then been shuffled randomly, so it is impossible to reassemble them into coherent single documents or even to know where they originate from, especially given that these documents are not paginated. Merely as an example, IMP120517-18 presents two unpaginated pages of a document entitled "Strings Rapid Application Toolkit." IMP120519-IMP120527, however, then presents a series of random and unrelated and unpaginated pages containing computer code and blank pages. IMP120528 then picks back-up with a three page record, again unpaginated, whose relation to the earlier "Strings Rapid Application Toolkit" is unclear.

Second, Implicit's production format makes it impossible to tell the date of a great number of these documents because, in addition to the above, they have been produced as single page images with no date printed on the page and no meta-data.

Both of these problems seem to be caused by the fact that this production consists primarily of information imperfectly retrieved from thousands of documents on some internal Intranet. As you know, Federal Rule of Civil Procedure 34 requires Implicit to produce these documents "as they are kept in the usual course of business." Please confirm, therefore, that Implicit will produce the actual Intranet in its native format with all relevant links intact.

Implicit's Interrogatory Responses

Again, thank you for agreeing to supplement Interrogatories Nos. 2, 3, 5, 6 and 9. However, as I have repeatedly mentioned, these requests to Implicit have been pending for over a year, notwithstanding months of sustained efforts by Juniper to obtain the missing or withheld information from Implicit. Please confirm that we will have your supplemental responses by no later than March 8, 2012.

Juniper's Document Production

As was discussed on the meet-and-confer, Juniper's production is the result of a reasonable search for responsive documents, and we disagree with Implicit's assertion that Juniper's "technical production" is incomplete. Nevertheless, as a sign of good faith and in the spirit of avoiding unnecessary disputes, we will agree to conduct a further review of our production and attempt to determine whether

IRELL & MANELLA LLP

A REGISTERED LIMITED LIABILITY LAW PARTNERSHIP
INCLUDING PROFESSIONAL CORPORATIONS

Spencer Hosie
March 2, 2012
Page 4

additional documentation may be available, in which case we will supplement our production.

With respect to your complaints regarding the search terms Juniper used, I again note that Juniper disclosed the fact that it was intending to use search terms and the specific terms that would be used on October 20, 2011, **over four months ago**. Implicit never once complained. Indeed, it was only after I sent my letters of January 23 and February 4, 2012, identifying numerous deficiencies in Implicit's discovery responses (issues which remain outstanding to this day) that Implicit raised this "issue" for the first time.

Juniper's Interrogatory Responses

Implicit claimed earlier that Juniper has been involved in "scores" of patent litigations to which it should have licenses, and we are still awaiting your list of the "scores" of patent cases you had in mind. Implicit has failed to explain what particular licenses it believes are missing, and has instead simply cited to a number of litigations in which Juniper was a party, without any regard as to the status of the case (a number of them are still pending) or the outcome. Nevertheless, we will review the information you have provided and supplement our production as necessary.

Sincerely,

/s/ Nima Hefazi
Nima Hefazi

NH

EXHIBIT 13



JUNOS OS: THE POWER OF ONE OPERATING SYSTEM

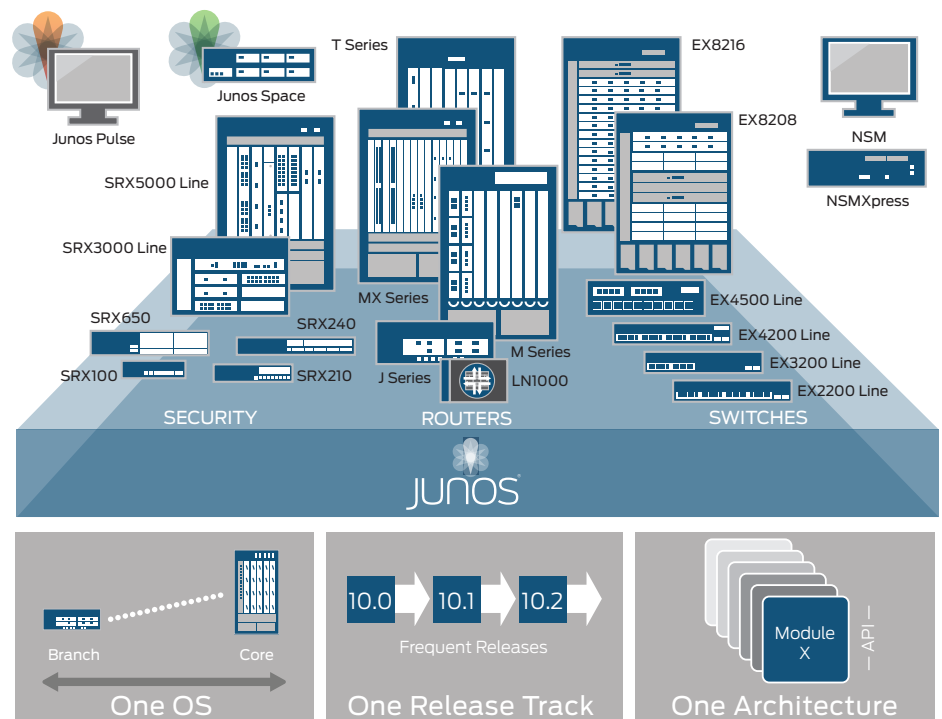
Reduce Complexity to Increase Availability and Deploy Services Faster with Lower TCO

Overview

Juniper Networks® Junos® operating system provides a common language across Juniper's routing, switching and security devices. The power of one Junos OS reduces complexity in high-performance networks to increase availability and deploy services faster with lower TCO.

What sets Junos OS apart from other network operating systems is the way it is built—one operating system delivered in one software release track and with one modular architecture.

The consistent user experience and automated toolsets of Junos makes planning and training easier, day-to-day operations more efficient, and changes faster in the network. Further, one operating system integrating new functionality in software protects customer investment, not only in hardware, but also in internal systems, practices, and knowledge. And that means not only lower TCO, but also greater flexibility in meeting the new needs and opportunities of the business.



Increasing Demands on the High-Performance Network

The network fundamentally runs the operations of high-performance enterprise and service provider businesses. Complex networks that require extensive rework to scale and change can slow down marketplace response and new business initiatives.

While old hardware and outdated or poorly integrated technologies present challenges, it is the software running in IP networks that consumes the most operational time, causes the majority of operational headaches, and creates obstacles to change. Largely based on source code initially built decades ago, legacy network software carries a number of limitations, including:

- **Complex, error-prone administration tasks**, which add not only time and effort to routine activities but also multiply the risk of human error that can lead to outages or create security vulnerabilities.
- **Multiple release trains and software versions**, which slow down network upgrades with requirements for extensive testing, qualification, and training while impacting the predictable delivery of new service features and fixes.
- **Monolithic software architectures**, which impact network stability, performance, and security with comingled operating system processes vying for the same shared computing resources, and where even a small problem in one process can cascade to affect many others.

So, how can you develop a network that cost-effectively scales with traffic growth, adapts along with changing business needs, and delivers new services, all while maintaining the operational stability of your infrastructure?

The solution begins with greater confidence in the underlying network foundation. If you can trust the software supporting your infrastructure, particularly in its most strategic and distributed components, your team can focus more of its time and effort keeping up with traffic demand as well as new application and business requirements.

Junos OS: The Foundation of High-Performance Networks

The Junos operating system provides a common language across Juniper's routing, switching and security devices and that reduces the complexity of not only the network design, but also its operation.

Different by Design

The key advantages of Junos OS derive primarily from how it is built—what Juniper calls the power of one differences:

- One operating system across all types and sizes of platforms reduces the time and effort to plan, deploy, and operate network and security infrastructure.
- One release track meets changing needs in software with stable delivery of new functionality in a steady, time-tested cadence.
- One modular software architecture provides highly available, secure and scalable software open to automation and partner innovation.

One Operating System

The truly unique nature of Junos OS begins with its most fundamental virtue: a single source code base. This means that Juniper Networks engineers can develop new features one time and then share the code, as applicable, across the many platforms running Junos OS.

A single, cohesive operating system providing a consistent user experience makes planning easier, day-to-day operations more intuitive, and changes faster. Administrators can configure and manage functionality from the basic chassis to complex routing using the same tools across devices to monitor, manage, and update the entire network. Juniper Networks Junos Space provides one system to manage security, switching, and routing platforms. Inherent interoperability simplifies new feature deployment, software upgrades, and other modifications, allowing operations teams to function more efficiently with less training time and lower costs.

One Software Release

Juniper builds Junos OS along a single “release train” — a disciplined plan for development with strict engineering principles that include rigid quality metrics and testing. Juniper does not replicate or recreate code to form multiple software trains or many different sets of feature packages as is the standard practice for other vendors. Rather, new releases build-up on the prior, creating the single release train delivered in a series of numbered versions. The Juniper Networks approach to software development produces a stable code base that not only reduces the number of unplanned system events, but also the time and trouble of planned maintenance and upgrades.

In over eleven years of development, Juniper has delivered new releases of expanding functionality four times each year, year after year. Each new release supports each product family for its role and application in the network. Whenever you are ready to upgrade, you simply choose and qualify a higher release number than your current version. Juniper provides over three years of support for its extended end-of-life releases. Customers count on the reliability and predictable behavior of the single Junos OS release train and confidently upgrade when they want to enable new functionality in their network.

One Modular Software Architecture

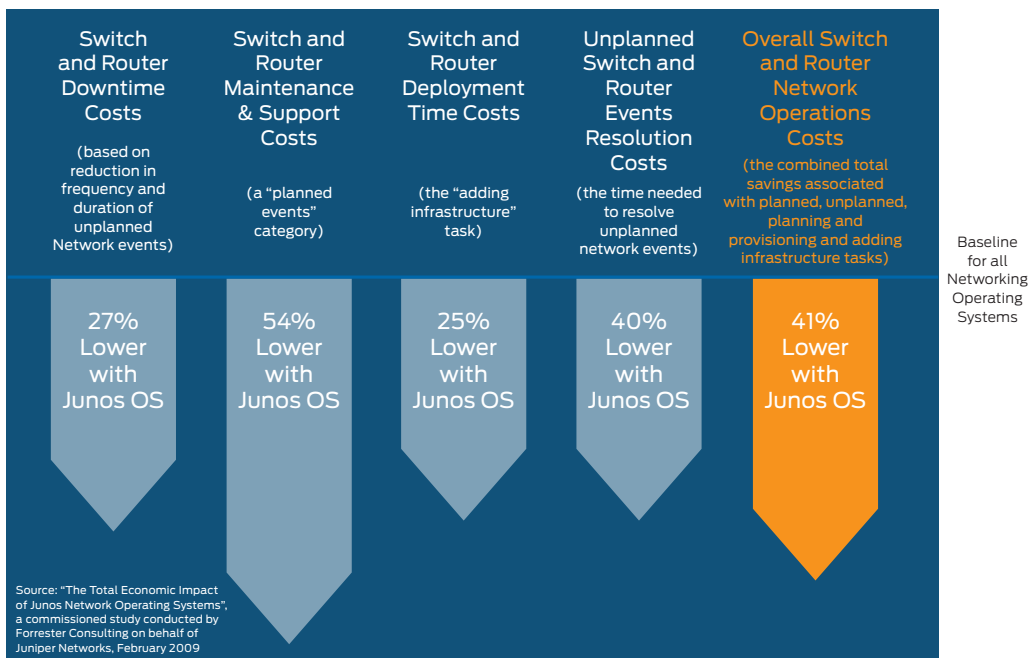
The software architecture of Junos OS is a modular design conceived for flexible, yet stable, innovation across many types of networking functions and sizes of platforms. Modularity and well-defined interfaces throughout the architecture streamline new development and enable complete, holistic integration of services. Through the delivery of one operating system that meets an expanding set of integrated requirements, customers can utilize hardware that can be incrementally expanded to support new growth and services for years to come. The approach extends customer investment not only in devices, but also in their internal systems, practices, and knowledge.

The advantages of modularity reach beyond the stable, evolutionary design of the software. For example, the process modules of the architecture run independently in their own protected memory space, so one module cannot disrupt another by scribbling on its memory. And, the architecture provides separation between control and forwarding functions to support predictable high-performance with powerful scalability from small to very large platforms. The modularity of the Junos OS architecture is thus integral to the high reliability, performance, and scalability delivered by its software design.

Delivering a High-Performance Network Foundation

Propelled by the power of one differences, Junos OS has rapidly evolved over the years in many dimensions to accommodate increasingly complex application and service needs. Juniper Networks platforms simultaneously scale integrated security and networking capabilities without compromising high performance and reliability.

Junos OS helps customers to save time and lower costs by reducing operational challenges and improving operational productivity. A commissioned study conducted by Forrester Consulting on behalf of Juniper Networks, The Total Economic Impact™ of Juniper Networks' Junos network operating system, examines the total economic impact and potential return on investment (ROI) enterprises may realize by deploying Junos OS in an enterprise network environment. (see Figure 1). Among other top-line results, the independent study found that interviewed companies, through the use of Junos OS and Juniper switches and routers, achieved a 40% reduction in operations costs for certain network operations tasks including planning and provision, deployment, and planned and unplanned network events.



Deploying routing, switching, and security platforms run by Junos OS deliver three key advantages to your networking infrastructure:

Continuous Systems: Improve network availability and the delivery of applications and services through high-performance software design, high availability features, prevention of human errors, and proactive operations measures.

Automated Operations: Increase productivity to lower operational expenses by reducing complexity with time saving configuration, automation of operations tasks, and centralized management.

Open Innovation: Enhance flexibility to deliver new services and applications, including secure interfaces and tools that open development to partners and customers for developing and deploying onboard applications on the Junos OS.

Continuous Systems

The consequence of an outage in a modern multiservice network can be extraordinarily expensive in terms of lost customer connections and transactions, as well as damaged customer confidence and penalties. Many different types of events and errors can cause disruption to network availability. Network equipment downtime can come from planned maintenance activities, unplanned hardware or software events, and most often according to many different studies, human error.

Addressing downtime, therefore, requires a multifaceted design approach that proactively considers all underlying factors. Devices running Junos OS have a well-deserved reputation for continuous performance and operational stability. The engineering foundations of continuous systems are rooted in the long standing design and software development philosophies of Junos OS; this is not a feature or attribute that can be easily retrofitted. Junos OS functionality for high availability includes expected failover and other service mechanisms, along with a range of capabilities unique to Juniper Networks, such as our disciplined processes for software development, error-resilient configuration, unified in-service software upgrade (ISSU), and automation of technical support services, among others.

Tools for automating operations are essential to maintaining high uptime. They not only reduce the severity and duration when unplanned network events do occur, but also can proactively prevent events from even happening, as discussed in the next section.

Automated Operations

The operational benefits of Junos OS derive not only from the reliability, performance, and security of its design, but also from a dedicated focus on simplified, error-resilient tasks across all operations functions. The hindsight that comes from prior experience has helped Junos OS engineers find better ways to design operations steps, interfaces, and tools. Many of these improvements simplify operations and reduce human error through increased automation.

Configuration

The Junos OS command-line interface (CLI) is easy to learn, with a feel that is similar to other command sets. Prominent improvements over other systems include error-resilient configuration with changes posted to a candidate file, flexible editing with time-saving shortcuts, automated checks of configurations, version control and rollback flexibility to restore prior configurations, and automated rollback in systems inadvertently isolated by configuration changes.

The most frustrating of human errors are ones that have happened before because they are repeating known mistakes that operations teams could ideally prevent. Junos OS configuration automation directly addresses this challenge through the customization of the commit verifications that run before a configuration becomes active. A library of scripts can be developed and maintained by your most experienced engineers to ensure that configurations are compliant with your business, network, and security policies. Moreover, these advanced tools include a macro capability that can condense repeated complex configurations into only a few configuration lines and variables.

Monitoring, Troubleshooting, and Problem Resolution

While network operations teams frequently spend their time in reactive mode, proactive discovery of potential issues is the preferred approach. Extensive monitoring and instrumentation capabilities within Junos OS give operations teams broad visibility into system health and device performance, along with the operational status of the network.

One of the characteristics of complex systems is the cascade effect of issues, with small problems capable of rapidly escalating into major ones. Junos OS event automation allow network and security engineers to automate early warning systems that not only detect emerging problems, but can also take immediate steps to avert further issues and restore normal operations. Your operational procedures can be captured in on-box command sets, not just paper, leveraging expertise across your company. The approach enables a continuous improvement capability as each outage and issue is diagnosed and proactive avoidance steps are written by your top engineers.

Open Innovation

Juniper Networks has extensively adopted and promoted open standards and interfaces for customers to manage and operate its networking and security platforms in multivendor networks. Junos OS provides multiple open interfaces, such as RADIUS, NETCONF/XML and DMI, for policy control, network management, and integration to other operations systems. The time tested interoperability and integration capabilities of Junos OS are evident in deployments in the largest service providers worldwide, and in tens of thousands of enterprise and government networks.

The Junos operating system also offers a software development platform that allows customers and partners to develop and deploy their own unique applications. The Junos SDK provides the necessary tools, libraries, and interfaces to build secure applications that run on Junos OS. In its over two years of availability, Junos SDK application development has grown with partners such as Advanced Broadband Networks, Harris Stratex, Lockheed Martin, NEC, NTT, Telchemy, and Triveni Digital developing a diverse mix of network applications.

Junos Platform

The Junos operating system is a part of the Junos Platform, an open software platform enabling dynamic, network-aware applications that interact with the network from the client to the cloud. With the Junos Platform, Juniper's customers can expand network software and interfaces to the application space, deploy software clients to control delivery, and accelerate the pace of innovation with an ecosystem of developers.

The Junos Platform provides customers and third-party developers unmatched flexibility to build applications by providing development interfaces at multiple layers of the network: in the networking device, across the network application layer, and at the network client. Unlike other platforms that merely enable third parties to integrate through APIs, the Junos Platform provides a true development environment including SDKs to create applications. The components of the Junos Platform are the Junos operating system, the Junos Space network application platform, and the Junos Pulse integrated network client.

Junos operating system includes an SDK for the development of onboard applications that deliver new control and packet processing functionality. The intelligent and secure interfaces of the Junos SDK give developers powerful options to build applications leveraging the underlying control and packet processing functionality of the operating system.

Juniper Networks Junos Pulse is a standards-based, dynamic, integrated multiservice network client for desktops, notebooks, netbooks, smartphones and other mobile and non-mobile devices, interfacing with integrated, multi-service network gateways to deliver identity- and location-aware anytime/anywhere connectivity, security, acceleration and collaboration with a simplified user experience while also serving as a development and integration platform for select third party applications.

Juniper Networks Junos Space is a programmable and extensible multipurpose Web 2.0 network application platform that enables the rapid development and deployment of applications to reduce cost and complexity, and to open the network to new business opportunities. Junos Space includes a core set of network infrastructure and operations management applications, such as Junos Space Ethernet Design, Network Activate, Route Insight, Virtual Control and Service Insight, for scaling services, simplifying operations and automating support.

Junos Ready Software is a growing portfolio of third-party and Juniper-developed applications based on the Junos operating system, Junos Space and Junos Pulse:

- **Dynamic Application Awareness** – Advanced router-integrated application identification and statistics collection helps customers maintain tight control over network resources at the Universal Services Edge.
- **Intrusion Prevention System (IPS)** – Tightly integrates Juniper Networks' latest and most advanced security features and provides protection from a wide range of threats and attacks in Universal Services Edge routers.
- **Media Flow Solution** – Scalable content delivery, distribution, and caching solution is purpose-built to address service providers' needs for a new media network. It combines intelligent, content-aware software running on high-performance hardware.
- **Session Border Control (SBC)** – Router-integrated session border control (SBC) products include a Border Signaling Gateway, a Border Gateway Function, and an Integrated Multiservice Gateway (IMSG) solution for routers at the Universal Services Edge to ensure the appropriate handling and quality delivery of real-time services over converged IP networks.
- **StreamScope eRM** – Comprehensive router-integrated video monitoring and analysis improves service availability and quality, while reducing operational complexity by eliminating standalone video probes in MX Series routers.
- **Telchemy ePM** – Sophisticated router-integrated, performance-monitoring application provides detailed analysis for a wide variety of IP services, including voice over IP (VoIP) in various edge and core routers.
- **Webroot** – Detection engine for malware protection in Junos Pulse enables customers to restrict access to corporate data and applications when security threats are detected, and to automatically eradicate threats so that mobile employees can stay connected and productive.

Portfolio of Platforms

Juniper Networks drives Junos OS innovation through its disciplined development as one network operating system. Juniper solutions provide consistency and reliability with routing, switching, and security platforms run by the same operating system across the high-performance network infrastructure. Our extensive portfolio connects branch, and regional offices, central sites and data centers, along with the metro, edge, and core sites of service provider networks. Juniper is leveraging its heritage of best-in-class services and security technology by delivering a broad set of intelligent and dynamic services in the Junos OS for security, broadband, voice, and video.

Routing, Switching, and Security

Juniper Networks EX Series Ethernet Switches address the access, aggregation, and core layers of branch office, campus, and data center networks, lowering operational expenses, including recurring power and cooling costs. The switches reduce capital expenses through innovative virtualization capabilities by collapsing network layers and reducing the amount of needed devices. The EX Series switches meet today's most advanced switching requirements for security and unified communications with integrated access control policy enforcement and extensive quality of service (QoS) features.

Juniper Networks J Series Services Routers offer predictable high performance and a variety of flexible interfaces that deliver secure, reliable network connectivity to remote, branch, and regional offices. The J Series consolidates market-leading routing, security, application optimization, and voice capabilities onto a single, easy-to-manage platform with options that include integrated Juniper Networks WXC Series Application Acceleration Platforms and integrated voice gateway technology from Avaya.

Juniper Networks JCS1200 Control System is the industry's first high-performance control plane scaling system. JCS1200 introduces independent scale of control and forwarding plane resources to maximize service growth, operational efficiencies, and control. This unique architecture enables service providers to rapidly expand their service offerings, and helps to reduce capital and operating expenditures.

Juniper Networks LN1000 Mobile Secure Router is an edge access router that delivers high-performance routing, firewall and intrusion prevention system technology (IPS) in a small form factor, energy-efficient package. Designed for the portable and transportable router markets, with the option to embed in a customer chassis, it's ideally suited to the most demanding mobile network applications.

Juniper Networks M Series Multiservice Edge Routers, spanning from 7 to 320 Gbps of throughput, uniquely combine best-in-class IP/MPLS capabilities with unmatched reliability, stability, security, and service richness. These multiservice edge routing platforms—deployed predominantly at the service-provider edge and in large, high-performance enterprise applications—enable consolidation of multiple networks onto a single IP/MPLS infrastructure without performance or feature compromise.

Juniper Networks MX Series 3D Universal Edge Routers, spanning from 80 Gbps to 2.6 Tbps of throughput, establish a new industry standard for Ethernet capacity, density, and performance. Offering efficient support of high-density interfaces and high-capacity switching throughput, the MX Series supports a wide range of business and residential applications and services, including high-speed transport and VPN services, next-generation broadband multiplay services, and high-volume Internet data centers.

Juniper Networks SRX Series Services Gateways secure enterprise and service provider infrastructure and applications with unrivaled performance and scalability. Based on the Dynamic Services Architecture, and engineered from the ground up to offer robust networking and security services, the SRX Series meets the network and security requirements of data center hyper-consolidation, rapid managed services deployments, and aggregation of security solutions.

Juniper Networks T Series Core Routers, spanning from 320 Gbps to 25 Tbps of throughput, provide high availability, reliability, performance, and scale, reducing operational and capital costs. The T Series offers sophisticated processing capabilities on a true multiservice platform with seamless integration with optical transport networks. Building core next-generation networks with T Series Core Routers offers a “pay-as-you-grow” path. Providers can reduce operational and capital expenses while customizing the network solution set and user experience.

Management, Identity and Policy, and Support Applications

Juniper Networks provides tools to centrally manage and support networking infrastructure. These products bring new capabilities to network and security management, and include a rich set of features that provide greater control for rapidly creating and deploying new IP services.

Juniper Networks Advanced Insight Solutions deliver a comprehensive set of tools and technologies to automate the delivery of network and device information for proactive network protection and support services offered by the Juniper Networks Technical Assistance Center (JTAC).

Juniper Networks J-Web is a Web-based GUI that provides users with simple to use tools to administer and manage Junos OS, including configuration, monitoring, and troubleshooting functions.

Juniper Networks Junos Scope software includes monitoring, configuration, inventory, and software management applications for managing IP services for the J Series, M Series, MX Series, and T Series routing platforms.

Juniper Networks Junos Space includes applications for network infrastructure automation with new management modules offered in each new release, such as:

- **Ethernet Design** – automates configuration, monitoring and administration of switch and router networks for campus and data center environments
- **Network Activate** – automates network discovery, design, deployment and validation for CE service provisioning
- **Route Insight** – audit-level visibility, troubleshooting and change modeling for L3 MPLS/IP networks
- **Security Design** – automates device and security service deployment, and policy lifecycle management
- **Service Insight** – proactive network state and device assessment against known issues
- **Service Now** – automates fault and case management
- **Virtual Control** – provides unified management of virtual and physical networks

Juniper Networks Network and Security Manager provides an easy-to-use solution that controls all aspects of firewall/VPN security platforms, J Series, M Series, and MX Series routing platforms, EX Series Ethernet Switches, SA Series SSL VPN Appliances, IC Series Unified Access Control Appliances, and IDP Series Intrusion Detection and Prevention Appliances, including device configuration, network settings, and security policy management.

Juniper Networks SDX300 Service Deployment System is a robust, customizable application that makes it possible for service providers to rapidly create and deploy new IP services to hundreds of thousands of subscribers.

Juniper Networks SRC Series Session and Resource Control Modules provide key policy and control layer functions including policy management, subscriber management, and authentication, authorization, and accounting (AAA), as well as network resource control.

Juniper Networks Steel-Belted Radius (SBR) Servers Enterprise Series provide uniform security policy enforcement across all network access methods, including WLAN, remote/VPN, dial, and identity-based (wired 802.1X). **Steel-Belted Radius Servers Service Provider Edition** servers support a broad range of network access and subscriber management database environments.

Juniper Networks STRM Series Security Threat Response Managers combine, analyze and manage an incomparable set of surveillance data --network behavior, security events, vulnerability profiles and threat information --to empower companies to efficiently manage business operations on their networks from a single console.

Solution Planning, Implementation, and Deployment

Getting Started with Junos OS

Adoption of any new product or technology initially requires some effort; however, our customers have consistently found the initial short-term activities of Junos OS adoption to be far outweighed by the long-term benefits. As a Juniper Networks customer, you have available all of the tools you will need to make the migration to Junos OS simple and safe, from the inherent characteristics of Junos OS itself to a wealth of education and support services.

Juniper Networks Education Services

Certified networking and security professionals are in greater demand than ever before, adding value to your organization through their skilled knowledge, particularly when that knowledge extends across multiple vendors to design best-in-class solutions for your organization. Juniper provides a wide array of training programs and a range of technical certifications. See the complete list of the Junos OS training and certifications at: www.juniper.net/training.

For teams new to Junos OS, the Juniper Networks Technical Certification Program (JNTCP) allows participants to gain practical competence with Junos OS deployment and operations. The Juniper Networks Certification Fast Track Program for enterprises significantly reduces the time and costs of training and certification for experienced networking and security professionals with existing routing, switching, and security knowledge. Find out more at: www.juniper.net/training/fasttrack.

Junos OS Offers the Power of One Operating System

Junos OS is a single network operating system providing a common language across Juniper's routing, switching and security devices. The power of one Junos OS reduces complexity in high-performance networks to increase availability and deploy services faster with lower TCO. Find out more by visiting: www.juniper.net/junos.

About Juniper Networks

Juniper Networks, Inc. is the leader in high-performance networking. Juniper offers a high-performance network infrastructure that creates a responsive and trusted environment for accelerating the deployment of services and applications over a single network. This fuels high-performance businesses. Additional information can be found at www.juniper.net.

JUNIPER NETWORKS SERVICE AND SUPPORT

Juniper Networks is the leader in performance-enabling services and support, which are designed to accelerate, extend, and optimize your high-performance network. Our services allow you to bring revenue-generating capabilities online faster so you can realize bigger productivity gains and faster rollouts of new business models and ventures. At the same time, Juniper Networks ensures operational excellence by optimizing your network to maintain required levels of performance, reliability, and availability. For more details, please visit www.juniper.net/us/en/products-services/.

Corporate and Sales Headquarters

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
Phone: 888.JUNIPER (888.586.4737)
or 408.745.2000
Fax: 408.745.2100
www.juniper.net

APAC Headquarters

Juniper Networks (Hong Kong)
26/F, Cityplaza One
1111 King's Road
Taikoo Shing, Hong Kong
Phone: 852.2332.3636
Fax: 852.2574.7803

EMEA Headquarters

Juniper Networks Ireland
Airside Business Park
Swords, County Dublin, Ireland
Phone: 35.31.8903.600
EMEA Sales: 00800.4586.4737
Fax: 35.31.8903.601

Copyright 2010 Juniper Networks, Inc. All rights reserved. Juniper Networks, the Juniper Networks logo, Junos, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

 Printed on recycled paper

EXHIBIT 14

Implicit Networks, Inc. v. Juniper Networks, Inc.

Oliver Tavakoli - CONFIDENTIAL

UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

IMPLICIT NETWORKS, INC.,)	
)	
Plaintiff,)	
)	
vs.)	No. C 10-4234 SI
)	
JUNIPER NETWORKS, INC.,)	
)	
Defendant.)	
_____)	

HIGHLY CONFIDENTIAL - ATTORNEYS' EYES ONLY
DEPOSITION OF: OLIVER TAVAKOLI
TAKEN ON: June 19, 2012

13145

BRENDA L. MARSHALL
CSR No. 6939

Implicit Networks, Inc. v. Juniper Networks, Inc.

Oliver Tavakoli - CONFIDENTIAL

		Page 149
1	mean, SMTP is one, but if you think of Google	13:41:13
2	mail and Yahoo! mail and attachments you might	13:41:16
3	have on top of that, that doesn't go by SMTP.	13:41:19
4	That goes by HTTP.	13:41:22
5	So I wouldn't -- I mean, what	13:41:24
6	constitutes layer 7 in this day and age is kind	13:41:27
7	of amorphous, I would say, at best.	13:41:30
8	MR. HOSIE: Okay. If I can have this	13:41:36
9	marked as the next in order. Five? Four?	13:41:37
10	THE REPORTER: Four.	13:41:45
11	(Whereupon, the document referred to	13:41:47
12	was marked Plaintiff's Exhibit 4 for	
13	identification by the Reporter, a	
14	copy of which is attached hereto.)	13:41:48
15	BY MR. HOSIE:	13:41:48
16	Q. For the record, I've had marked as	13:41:53
17	Exhibit 4 a portion of a claims chart captioned	13:41:55
18	"Implicit Networks, Inc., Flow Based Processing,	
19	'163 C1 Patent Claim 1."	13:42:03
20	Have you seen this document before, sir?	13:42:04
21	A. No, I have not.	13:42:05
22	Q. All right. Do you know what a claim	13:42:06
23	chart is?	13:42:08
24	A. No.	13:42:09
25	Q. Do you have any views on whether the	13:42:11

Page 150

1 Juniper SRX 5800 box infringes the Implicit 13:42:14

2 Network patents? 13:42:19

3 A. Having not read the patents, I don't 13:42:20

4 have any view on it. 13:42:22

5 Q. One way or the other? 13:42:23

6 A. I have no information. So -- 13:42:25

7 Q. So no view? 13:42:29

8 A. No view. 13:42:30

9 Q. All right, sir. Turn to the second 13:42:31

10 page, please. I can represent to you this is a 13:42:33

11 graphic taken from a Juniper document. Does it 13:42:36

12 look familiar to you? 13:42:38

13 A. Just a second. Not specifically, but, 13:42:39

14 you know, I have -- I have seen diagrams similar 13:42:49

15 to this. 13:42:51

16 Q. And -- and Juniper talks about one OS. 13:42:51

17 What does it mean by that? 13:42:58

18 MR. KAGAN: Objection. Lacks 13:43:00

19 foundation. Calls for speculation. 13:43:01

20 THE WITNESS: What Juniper means by one 13:43:04

21 OS is that there is a single code tree in which 13:43:07

22 all the source code for all the systems resides, 13:43:12

23 and that is all released on a fixed cadence. 13:43:15

24 What -- 13:43:19

25 BY MR. HOSIE: 13:43:20

Page 151

1 Q. Once a quarter? 13:43:20

2 A. It used to be once in a quarter, but the 13:43:21
3 cadence has changed. 13:43:23

4 What it does not mean is that all of 13:43:27

5 that software is loaded on each and every 13:43:29

6 system. It simply means that when you run Junos 13:43:32

7 10.4, you're getting stuff off of that same code 13:43:36

8 tree. And on system A versus system B versus 13:43:39

9 system C, depending on what functionality you 13:43:42

10 have, you have appropriate components for that 13:43:44

11 hardware to the functionality of that box needs. 13:43:47

12 And the best example of this is our 13:43:51

13 security functions are not -- even though we 13:43:53

14 have one Junos, our EX switches do not have our 13:43:56

15 security functions on them. 13:43:59

16 Q. So different products have different 13:44:00

17 functionality enabled, different portions of the 13:44:02

18 one Junos code tree? 13:44:05

19 A. Correct. 13:44:07

20 Q. Okay. If you turn to page 7, please. 13:44:07

21 This purports to be a methodological, 13:44:36

22 step-by-step walkthrough of creating a new data 13:44:39

23 processing path, based on information in the 13:44:41

24 first packet. I believe this is all going to be 13:44:43

25 implementation detail and something you're not 13:44:46

Page 152

1 familiar with, but let me ask. 13:44:49

2 Step 1, "Determine," and there's a site 13:44:50

3 source code, "number of plugins to use and 13:44:53

4 plugin map based on protocol of 'Flow 13:44:56

5 Selector.'" 13:44:58

6 Again, you don't know anything about how 13:44:59

7 that works? 13:45:00

8 A. This is simply the same words as were in 13:45:01

9 the diagram previously shown to me. So -- 13:45:03

10 Q. Okay. So I'd get the same series of 13:45:04

11 answers? 13:45:06

12 A. Yes. 13:45:06

13 MR. KAGAN: Although a picture is worth 13:45:08

14 a thousand words. 13:45:09

15 MR. HOSIE: We try. We try. 13:45:11

16 THE WITNESS: Mosaic 500. 13:45:12

17 BY MR. HOSIE: 13:45:14

18 Q. Okay. Going back to the SRX discussion, 13:45:14

19 you mentioned that there were three basic 13:45:18

20 configurations, one for the 5000 boxes, one for 13:45:20

21 the 1 to 3000 boxes, and one for the sub 1000 13:45:23

22 boxes. Let's talk about the first -- the -- the 13:45:27

23 1 to 1000 configuration setup. 13:45:31

24 A. The 1 to 1000 -- what do you mean -- 13:45:34

25 MR. KAGAN: No. 13:45:36

1
2
3 I, BRENDA L. MARSHALL, Certified
4 Shorthand Reporter, License No. 6939, do hereby
5 certify:

6 That, prior to being examined, the
7 witness named in the foregoing deposition, to
8 wit, OLIVER TAVAKOLI, was by me duly sworn to
9 testify the truth, the whole truth and nothing
10 but the truth:

11 That said transcript was taken down by
12 me in shorthand at the time and place therein
13 named and thereafter reduced to computerized
14 transcription under my direction.

15
16 I further certify that I am not
17 interested in the event of the action.

18
19
20 WITNESS this 3rd day of July, 2012.

21
22
23 _____
24 BRENDA L. MARSHALL
25

EXHIBIT 15

H I G H L Y C O N F I D E N T I A L

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

IMPLICIT NETWORKS, INC.,)	
)	
Plaintiff,)	
)	
v.)	Case No. C 10-4234SI
)	
JUNIPER NETWORKS, INC.,)	
)	
Defendant.)	
_____)	

H I G H L Y C O N F I D E N T I A L

DEPOSITION OF: STEFAN DYCKERHOFF

TAKEN ON: July 27, 2012

ROBERT R. CUDA
CSR NO. 2652

18

1 A. The conversation that I have with the customers 10:23:58
2 is about the business benefits they derive from Junos, 10:24:01
3 and it really focusses on how they manage their network. 10:24:04
4 I certainly don't go in and talk to them about the bits 10:24:08
5 and bytes of the architecture and how something is built 10:24:12
6 or not built. 10:24:14

7 BY MR. BISHOP: 10:24:16

8 Q. Okay. Well, let me be clear. In that question 10:24:16
9 when I said "you," I meant Juniper. Juniper tells its 10:24:19
10 customers, does it not, that the one Junos message 10:24:23
11 includes the benefit of one architecture; correct? 10:24:26

12 MR. KAGAN: Objection, vague, lacks foundation, 10:24:29
13 calls for speculation, compound. 10:24:32

14 A. So, you know, it could very well be that that's 10:24:35
15 in our marketing material. I have certainly seen, you 10:24:42
16 know, that every once in a while, but, again, it's not 10:24:44
17 what I do with the customer. That's not part of my job. 10:24:47

18 BY MR. BISHOP: 10:24:50

19 Q. Okay. But you have seen it in Juniper's 10:24:50
20 marketing material; correct? 10:24:54

21 A. Yes, I have seen that in some of our marketing 10:24:55
22 material. 10:24:58

23 Q. Okay. And Junos is one architecture, isn't it, 10:24:58
24 I mean, within the various products? 10:25:02

25 MR. KAGAN: Objection, vague. 10:25:04

19

1 A. I think what's important to understand about 10:25:06
2 Junos is that it allows customers to derive value from 10:25:11
3 it by having an interface. But Junos is a very big and 10:25:18
4 complex operating system with many differences between 10:25:21
5 the different products we have, and so, you know, what 10:25:26
6 exactly is shared or not, that's the engineer's job. 10:25:28
7 But if you look at our product portfolio, there's many 10:25:32
8 different things in there, and those work differently 10:25:35
9 inside of the code base. 10:25:38

10 BY MR. BISHOP: 10:25:39

11 Q. I am having trouble understanding. A minute 10:25:40
12 ago I believe you said that Junos is one architecture, 10:25:43
13 and I think you just said that actually the Junos code 10:25:45
14 base is quite different in the different products. Did 10:25:49
15 I understand that correctly? 10:25:52

16 A. Well, I think you asked me about marketing 10:25:53
17 material and whether we talk about the architecture and 10:25:57
18 how the customer interfaces with Junos, and there are 10:25:59
19 similarities there. But what I was pointing out is that 10:26:03
20 as the engineers make the choices for how to write 10:26:06
21 software for a given product, they certainly optimize 10:26:11
22 around that product. So there's many different 10:26:15
23 optimizations and hence different code inside the code 10:26:19
24 base. 10:26:19

25 Q. Would you say it's true or false that Junos has 10:26:20

20

1 one architecture for its various products? 10:26:23

2 MR. KAGAN: Objection, vague. 10:26:25

3 A. It's really not a question you can answer with 10:26:28

4 true or false, because it depends on what you care 10:26:32

5 about. Do you care about how you interface with the 10:26:35

6 product? Do you care about how it's built from the 10:26:37

7 bottom up? So it's kind of an impossible question to 10:26:40

8 answer. 10:26:43

9 BY MR. BISHOP: 10:26:45

10 Q. Okay. So you can't tell me whether that's true 10:26:45

11 or not? 10:26:47

12 MR. KAGAN: Objection, asked and answered. 10:26:48

13 A. No. 10:26:50

14 MR. BISHOP: Pardon me for taking a little 10:27:00

15 time. 10:27:03

16 All right. Let's mark this as Exhibit 1. It's 10:27:13

17 Juniper's 2011 10-K. 10:27:16

18 (Plaintiff's Exhibit 1 was marked for 10:27:20

19 identification) 10:27:50

20 Q. And, Mr. Dyckerhoff, this appears to be 10:27:50

21 Juniper's 10-K for 2011? I am not asking you to verify 10:27:54

22 every page. 10:27:58

23 A. Yeah, I can't say whether that's our entire 10:27:59

24 10-K or not, but certainly it says here it's the 10:28:03

25 Commission file number of the 10-K for Juniper Networks. 10:28:06

1 I, ROBERT R. CUDA, CSR No. 2652, certify:

2 That the foregoing deposition of STEFAN
3 DYCKERHOFF was taken before me at the time and place
4 therein set forth, at which time the witness declared
5 under penalty of perjury to tell the truth;

6 That the testimony of the witness and all
7 objections made at the time of the deposition were
8 recorded stenographically by me and were reduced to a
9 computerized transcript under my direction;

10 That this transcript is a true record of the
11 testimony of the witness and of all objections and
12 colloquy made at the time of the deposition;

13 I further certify that I am neither counsel
14 for nor related to any party to said action nor
15 interested in the outcome;

16 The certification of this transcript does not
17 apply to any reproduction of the same by any means
18 unless under the direct control and/or direction of the
19 certifying deposition reporter.

20 IN WITNESS WHEREOF, I have subscribed my name
21 this 4th day of August 2012.

22

23

24

ROBERT R. CUDA, CSR No. 2652

25

EXHIBIT 16



IDP Series

Concepts and Examples Guide

Release
5.1rX



Published: 2011-05-05
Revision 03

CHAPTER 1

IDP Series Product Overview

This chapter provides an overview of the intrusion detection and prevention (IDP) standalone solution and provides a documentation map of IDP features to IDP documentation sources. It includes the following topics:

- Juniper Networks IDP Solutions on page 3
- IDP Series Features Overview on page 3

Juniper Networks IDP Solutions

Juniper Networks provides **intrusion detection services and intrusion detection and prevention (IDP) technology** in the following product families:

- Juniper Networks IDP Series Intrusion Detection and Prevention Appliances
- Juniper Networks ISG Series Integrated Security Gateways
- Juniper Networks SRX Series Services Gateways

This guide describes the IDP Series appliances.

Related Documentation

The following related topic is included in the *IDP Series Concepts and Examples Guide*:

- IDP Series Features Overview on page 3

IDP Series Features Overview

Table 5 on page 3 briefly describes Juniper Networks IDP Series features.

Table 5: IDP Series Features

Feature	Description	Documentation
Application-Based Management		

- Configuring Advanced Settings for the User-Role-Based Policy Feature
- Verifying Integration with an IC Series Unified Access Control Appliance

The following related topic is included in the *IDP Series Deployment Scenarios*:

- Deploying IDP Series with an IC Series Device to Implement User-Role-Based Security Policies

Using Application Identification

The application identification feature enables the IDP engine to detect applications running on standard or nonstandard ports. Port-independent application identification enhances both security and manageability by eliminating the need to manually and comprehensively configure application-port mapping for the service objects and application objects used in the IDP rulebase and APE rulebase rules.

The application identification feature uses application signatures provided by the Juniper Security Center team (J-Security Center) to identify the session application. Beginning with IDP OS Release 5.1, the application identification feature can match extended application signatures used in APE rulebase rules. *Extended application* signatures are also called *nested application* signatures. The predefined extended application signatures developed for IDP OS Release 5.1 include the most prevalent Web 2.0 applications running over HTTP. If your security policy includes APE rules configured to match extended application signatures, the application identification process identifies and generates the following HTTP contexts: http-url-parsed, http-url-parsed-param-parsed, http-header-host, and http-header-content-type. The application identification feature can then match application signature patterns in those contexts.

J-Security Center updates application signatures and develops new ones as necessary. Beginning with IDP OS Release 5.1, you can use NSM to browse predefined application objects, predefined extended application objects, and application groups. You can also use NSM to create custom application definitions, if needed. You cannot, however, create custom extended application definitions.

When the application identification feature identifies a new application, it caches the result (the destination address, port, protocol, and service) to reduce processing for subsequent sessions. The application cache and extended application cache are maintained separately.

When the IDP engine processes security policy rules, it examines the session, beginning with the first packet, to identify a match. To match service or application, the IDP engine first compares the session against the application identification cache to identify the application. If the session does not match the application identification cache, the IDP engine processes the session against the application signatures. If the IDP engine is still unable to determine the application, it uses the standard application protocol and port.

In IDP rulebase rules, with application identification enabled, you set the service object in rules to **Default** to allow the application identification feature to identify the correct

service. If you set service to a specific service object, application identification is not applied and the rule is processed using the service object properties.

In APE rulebase rules, with application identification enabled, you set the service object in rules to **Default** and specify rules based on application or extended application. If you disable application identification and specify a match based on application, the IDP engine uses the standard application protocol and port for the application. If the application you are interested in is not listed, you can create a custom application object to match against application properties that you define.

The application identification feature is enabled by default, and we recommend you use this feature. To support lab experimentation and troubleshooting, you can disable application identification and extended application identification, and you can tune the following settings:

- Maximum number of sessions that utilize application identification
- Maximum memory used by application identification
- Maximum memory for saving TCP or UDP packets per session

For information on tuning these parameters, see the *scio const* reference page in the *IDP Series Administration Guide*.

Related Documentation

The following related topic is included in the *IDP Series Concepts and Examples Guide*:

- IDP Rulebase Example: Using Application Identification on page 107
- Using Application Objects on page 121
- Understanding the IDP Rulebase on page 91
- J-Security Center Updates Overview on page 21

The following related topics are included in the *IDP Series Administration Guide*:

- Application Objects Task Summary
- Specifying Rule Match Conditions (NSM Procedure)
- *scio const*

Using Attack Objects

If the session matches rule settings for source, destination, service, and VLAN tag ID, the IDP engine decodes the traffic and inspects the session packets for the attack objects specified in the rule. The following topics provide guidelines for using attack objects in IDP rulebase rules:

- Attack Objects Overview on page 98
- Understanding Predefined Attack Objects and Attack Object Groups on page 99
- Using Attack Object Groups on page 99
- Using Custom Attack Objects on page 100

EXHIBIT 17

(12) **EX PARTE REEXAMINATION CERTIFICATE (7567th)****United States Patent**
Balassanlan(10) **Number:** **US 6,629,163 C1**(45) **Certificate Issued:** **Jun. 22, 2010**(54) **METHOD AND SYSTEM FOR
DEMULTIPLEXING A FIRST SEQUENCE OF
PACKET COMPONENTS TO IDENTIFY
SPECIFIC COMPONENTS WHEREIN
SUBSEQUENT COMPONENTS ARE
PROCESSED WITHOUT RE-IDENTIFYING
COMPONENTS**(75) Inventor: **Edward Balassanlan**, Kirkland, WA
(US)(73) Assignee: **Implicit Networks, Inc.**, Bellevue, WA
(US)

6,101,320 A	8/2000	Schuetze et al.
6,104,704 A	8/2000	Buhler et al.
6,128,624 A	10/2000	Papierniak et al.
6,192,419 B1	2/2001	Aditham et al.
6,199,054 B1	3/2001	Khan et al.
6,212,550 B1	4/2001	Segur
6,222,536 B1	4/2001	Kihl et al.
6,246,678 B1	6/2001	Erb
6,356,529 B1	3/2002	Zarom
6,405,254 B1	6/2002	Hadland
6,574,610 B1	6/2003	Clayton et al.
6,651,099 B1	11/2003	Dietz et al.
6,785,730 B1	8/2004	Taylor
7,233,948 B1	6/2007	Shamoon et al.

Reexamination Request:

No. 90/010,356, Dec. 18, 2008

Reexamination Certificate for:

Patent No.: **6,629,163**
 Issued: **Sep. 30, 2003**
 Appl. No.: **09/474,664**
 Filed: **Dec. 29, 1999**

Certificate of Correction issued Dec. 2, 2003.

(51) **Int. Cl.**
G06F 13/00 (2006.01)
H04L 12/54 (2006.01)
H04L 12/56 (2006.01)

(52) **U.S. Cl.** **710/33; 710/1; 710/3; 710/20;**
710/38; 710/51; 370/401; 370/487; 370/498;
370/535; 370/536; 370/542

(58) **Field of Classification Search** None
 See application file for complete search history.

(56) **References Cited****U.S. PATENT DOCUMENTS**

5,392,390 A	2/1995	Crozier
5,627,997 A	5/1997	Pearson et al.
5,768,521 A	6/1998	Dedrick
5,848,415 A	12/1998	Guck
6,047,002 A	4/2000	Hartmann et al.

OTHER PUBLICATIONS

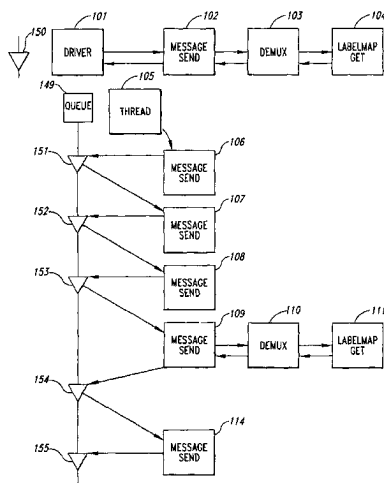
Internetworking with TCP/IP, vol. 1: Principles, Protocols, and Architecture, Second Edition, Douglas E. Comer, Prentice Hall, 1991, Chapter 10 and glossary; vol. II: Design, Implementation, and Internals, Douglas E. Comer and David L. Stevens, Prentice Hall, 1991, Chapters 1–3, 5, 10, 11, and 16 (Appendix “B” to Reexam).

TCP/IP Illustrated vol. 1, W., Addison-Wesley, 1994, Richard Stevens, chapters 1, 8, and 18; TCP/IP Illustrated vol. 2, Gary Wright and W. Richard Stevens, chapters 22, 24, 28, and 29, 1995 (Appendix “C” to Reexam).

Scout: A Path-Based Operating System, David Mosberger, 1997 (Doctoral Dissertation Submitted to the University of Arizona) (Appendix “D” to Reexam).

Primary Examiner—Matthew Heneghan(57) **ABSTRACT**

A method and system for demultiplexing packets of a message is provided. The demultiplexing system receives packets of a message, identifies a sequence of message handlers for processing the message, identifies state information associated with the message for each message handler, and invokes the message handlers passing the message and the associated state information. The system identifies the message handlers based on the initial data type of the message and a target data type. The identified message handlers effect the conversion of the data to the target data type through various intermediate data types.



US 6,629,163 C1

1

**EX PARTE
REEXAMINATION CERTIFICATE
ISSUED UNDER 35 U.S.C. 307**

THE PATENT IS HEREBY AMENDED AS
INDICATED BELOW.

Matter enclosed in heavy brackets [] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

Claims 1-5, 7, 9, 10, 12, 14-18, 20, 21, 23, 25, 26, 35-37, 39-41, 43 and 44 is determined to be patentable as amended.

Claims 6, 8, 11, 13, 19, 22, 24, 27-34, 38 and 42, dependent on an amended claim, are determined to be patentable.

New claim 45 is added and determined to be patentable.

1. A method in a computer system for processing a message having a sequence of packets, the method comprising: providing a plurality of components, each component being a software routine for converting data with an input format into data with an output format;

for the first packet of the message,

dynamically identifying a non-predefined sequence of components for processing the packets of the message such that the output format of the components of the non-predefined sequence match the input format of the next component in the non-predefined sequence, wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components after the first packet is received; and

storing an indication of each of the identified components so that the non-predefined sequence does not need to be re-identified for subsequent packets of the message; and

for each of a plurality of packets of the message in sequence,

for each of a plurality of components in the identified non-predefined sequence,
retrieving state information relating to performing the processing of the component with the previous packet of the message;
performing the processing of the identified component with the packet and the retrieved state information; and

storing state information relating to the processing of the component with packet for use when processing the next packet of the message.

2. The method of claim 1 wherein the storing of an indication of each of the *dynamically* identified components includes storing a key for use in retrieving state information relating to the message.

3. The method of claim 1 wherein a second component of the non-predefined sequence of components that are *dynamically* identified is identified after the processing of the first packet by a first component is performed.

4. The method of claim 1 wherein the packet may be transformed by each component of an identified non-predefined sequence.

2

5. The method of claim 1 wherein the identified non-predefined sequence of components for two messages are different.

7. The method of claim 6 wherein the identified non-predefined sequence of components for a message are executed by the thread for the message.

9. The method of claim 1 wherein the performing of the processing of the component includes deferring performing of the next component in the identified non-predefined sequence until multiple packets are processed by the component.

10. The method of claim 1 wherein the *dynamically* identifying of a non-predefined sequence of components includes deferring identification of the next component of the non-predefined sequence until processing of the last component identified so far in the non-predefined sequence is performed.

12. The method of claim 1 wherein an output format of a component in the identified non-predefined sequence for a message matches an input format of the next component in the identified non-identified sequence for the message.

14. The method of claim 1 wherein a plurality of non-predefined sequences of components are *dynamically* identified for a message.

15. A method in a computer system for demultiplexing packets of messages, the method comprising:

dynamically identifying a non-predefined sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components, wherein different non-predefined sequences of components can be identified for different messages, each component being a software routine, and wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components; and

for each packet of each message, performing the processing of the identified non-predefined sequence of components of the message wherein state information generated by performing the processing of a component for a packet is available to the component when the component processes the next packet of the message.

16. The method of claim 15 wherein the identified non-predefined sequence of components is identified as the first packet of the message is processed.

17. The method of claim 15 wherein a packet of a message processed by a component of the identified non-predefined sequence for the message is available to the next component in the identified non-predefined sequence.

18. The method of claim 15 wherein the components of an identified non-predefined sequence for a message are executed within a thread [associate] associated with a single message.

20. The method of claim 15 wherein the performing of the processing of the component includes deferring performing of the next component in the identified non-predefined sequence until multiple packets are processed by the component.

21. The method of claim 15 wherein the *dynamically* identifying of a non-predefined sequence of components includes deferring identification of the next component of the non-predefined sequence until processing of the last component identified so far in the non-predefined sequence is complete.

23. The method of claim 15 wherein an output format of a component in the identified non-predefined sequence for a message matches an input format of the next component in the identified non-predefined sequence for the message.

US 6,629,163 C1

3

25. The method of claim 15 wherein the identified *non-predefined* sequences of components are identified for a message.

26. A computer system for processing packets of messages, the [method] *system* comprising:

a plurality of components, each component having an input format and an output format;

identification means that identifies a sequence of components for each message after a packet of message has been received, such that the output format of a component in an identified sequence matches the input format of the next component in the identified sequence;

receiving means that receives packets of the messages; and

demultiplexing means that routes packets of messages to the sequence of components identified for each message for performing the processing of the components on the packets.

35. A computer-readable medium containing [instruction] *instructions* for demultiplexing packets of messages, by method comprising:

dynamically identifying a message-specific *non-predefined* sequence of components for processing the packets of each message upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific *non-predefined* sequence identified when the first packet was received, and wherein *dynamically identifying* includes selecting individual components to create the message-specific *non-predefined* sequence of components; and

for each packet of each message, invoking the identified *non-predefined* sequence of components in sequence to perform the processing of each component for the packet wherein each component saves message-specific state information so that that component can use the [save] *saved* message-specific state information when that component performs its processing on the next packet of the message.

36. The computer-readable medium of claim 35 wherein a second component of the message-specific *non-predefined* sequence is identified after the first packet is processed by a first component of the message-specific *non-predefined* sequence.

4

37. The computer-readable medium of claim 35 wherein a packet may be transformed by each component of an identified *non-predefined* sequence.

39. The computer-readable medium of claim 38 wherein the identified *non-predefined* sequence of components for a message is executed by the thread for the message.

40. The computer-readable medium of claim 35 wherein the performing of the processing of the component includes deferring performing of the next component in the identified *non-predefined* sequence until multiple packets are processed by the component.

41. The computer-readable medium of claim 35 wherein the *dynamically* identifying of a *non-predefined* sequence of components includes deferring identification of the next component of the *non-predefined* sequence until processing of the last component identified so far in the *non-predefined* sequence is performed.

43. The computer-readable medium of claim 35 wherein an output format of a component in the identified *non-predefined* sequence for a message matches an input format of the next component in the identified *non-predefined* sequence for the message.

44. The computer-readable medium of claim 35 wherein a plurality of *non-predefined* sequences of components are identified for a message.

45. A computer-readable medium containing instructions for demultiplexing packets of a message, by a method comprising:

identifying a message-specific sequence of components for processing the packets of each message upon receiving the first packet of the message wherein each component in the sequence is identified by using the output format of the previous component to identify a component with a compatible input format, and wherein subsequent packets of the message can use the message-specific sequence identified when the first packet was received;

for each packet of the message, invoking the identified sequence of components in sequence to perform the processing of each component for the packet, wherein each component saves message-specific state information so that that component can use the saved message-specific state information when that component performs its processing on the next packet of the message.

* * * * *

EXHIBIT 18

(10) **Patent No.:** US 7,711,857 B2
(45) **Date of Patent:** *May 4, 2010

(54) **METHOD AND SYSTEM FOR DATA DEMULTIPLEXING**

(75) Inventor: **Edward Balassanian**, Kirkland, WA
(US)

(73) Assignee: **Implicit Networks, Inc.**, Kirkland, WA
(US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 172 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: 11/933,022

(22) Filed: **Oct. 31, 2007**

(65) **Prior Publication Data**

US 2008/0133642 A1 Jun. 5, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/636,314, filed on Aug. 26, 2003, which is a continuation of application No. 09/474,664, filed on Dec. 29, 1999, now Pat. No. 6,629,163.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** 709/246; 709/236

(58) **Field of Classification Search** 709/230,
709/206, 246, 236; 370/231, 466
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,392,390	A	2/1995	Crozier	
5,627,997	A	5/1997	Pearson et al.	
5,768,521	A	6/1998	Dedrick	
5,848,415	A	12/1998	Guck	
6,047,002	A *	4/2000	Hartmann et al.	370/466

6,101,320	A	8/2000	Schuetze et al.	
6,104,704	A	8/2000	Buhler et al.	
6,128,624	A	10/2000	Papierniak et al.	
6,192,419	B1	2/2001	Aditham et al.	
6,199,054	B1	3/2001	Khan et al.	
6,212,550	B1 *	4/2001	Segur	709/206
6,222,536	B1	4/2001	Kihl et al.	
6,246,678	B1	6/2001	Erb	
6,356,529	B1 *	3/2002	Zarom	370/231
6,405,254	B1 *	6/2002	Hadland	709/230
6,574,610	B1	6/2003	Clayton et al.	
6,651,099	B1	11/2003	Dietz et al.	
6,785,730	B1 *	8/2004	Taylor	709/230
7,233,948	B1	6/2007	Shamoon et al.	

OTHER PUBLICATIONS

Douglas E. Comer, *Internetworking with TCP/IP*, vol. I: Principles, Protocols, and Architecture, Second Edition, Prentice Hall, 1991, Chapter 10 and Glossary.

(Continued)

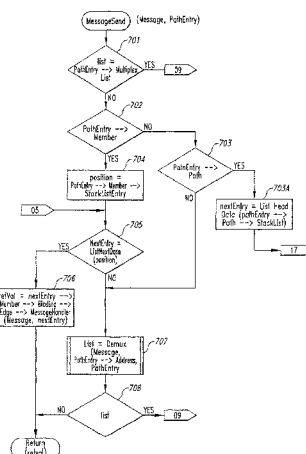
Primary Examiner—Jungwon Chang

(74) *Attorney, Agent, or Firm*—Gard & Kaslow LLP

(57) **ABSTRACT**

A method and system for demultiplexing packets of a message is provided. The demultiplexing system receives packets of a message, identifies a sequence of message handlers for processing the message, identifies state information associated with the message for each message handler, and invokes the message handlers passing the message and the associated state information. The system identifies the message handlers based on the initial data type of the message and a target data type. The identified message handlers effect the conversion of the data to the target data type through various intermediate data types.

10 Claims, 16 Drawing Sheets



US 7,711,857 B2

Page 2

OTHER PUBLICATIONS

Douglas E. Comer and David L. Stevens, Internetworking with TCP/IP, vol. II: Design, Implementation, and Internals, Prentice Hall, 1991, Chapters 1-3, 5, 10, 11 and 16.

W. Richard Stevens, TCP/IP Illustrated vol. 1, Addison-Wesley, 1994, Chapters 1, 8 and 18.

Gary Wright and W. Richard Stevens, TCP/IP Illustrated vol. 2, Addison-Wesley, 1995, Chapters 22, 24, 28 and 29.

David Mosberger, Scout: A Path-Based Operating System, Doctoral Dissertation Submitted to the University of Arizona, 1997, Appendix "D".

Dawson R. Engler and M. Frans Kaashoek; DPF: Fast, Flexible Message Demultiplexing using Dynamic Code Generation, published 1996 by the Association for Computing Machinery, Inc.

Maresh Jayaram and Ron K. Cytron; Efficient Demultiplexing of Network Packets by Automatic Parsing, published Jul. 1995 by Washington University of Computer Science.

Masanobu Yuhara, Brian N. Bershad, Chris Maeda, and J. Eliot B. Moss; Efficient Packet Demultiplexing for Multiple Endpoints and Large Messages, published 1994.

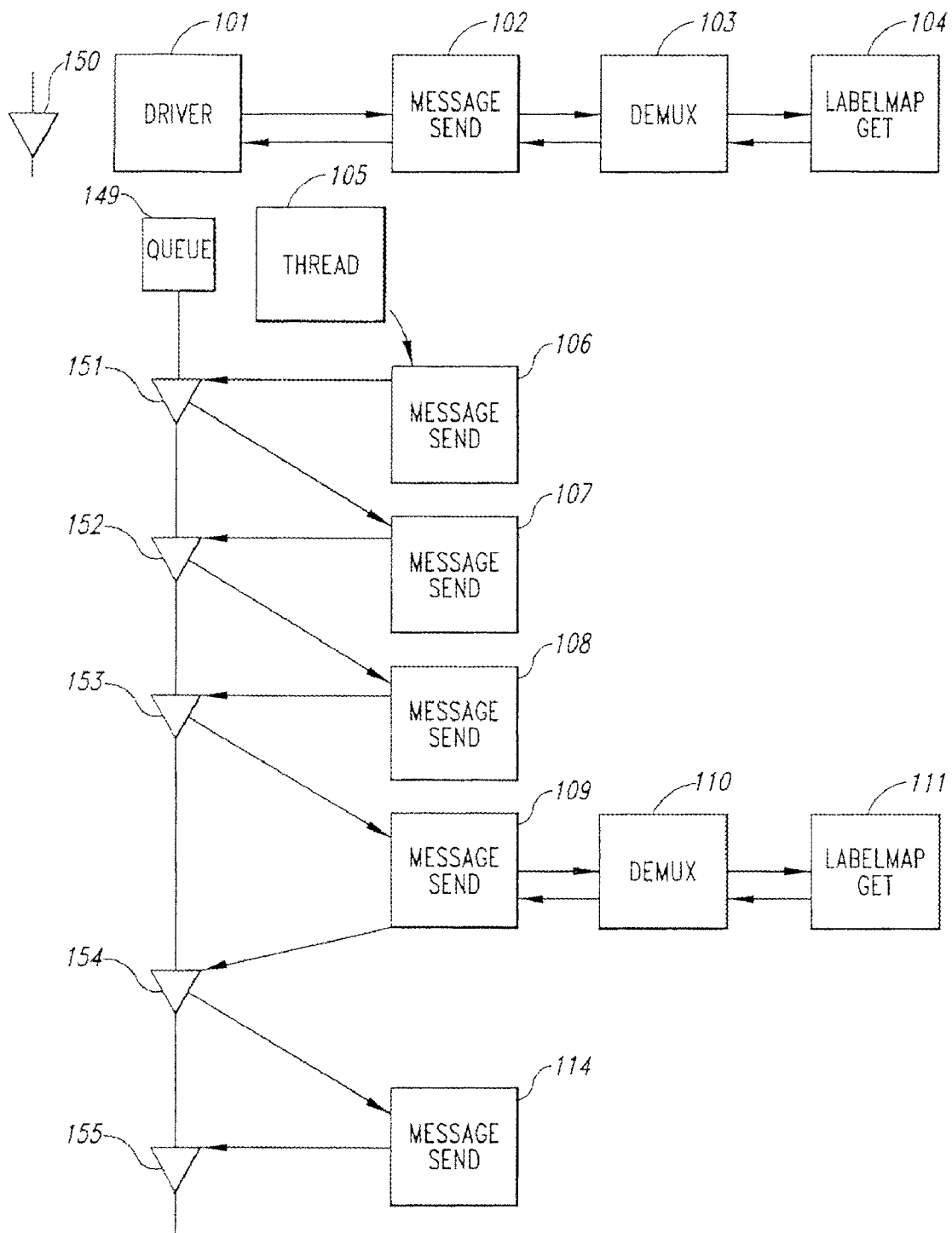
* cited by examiner

U.S. Patent

May 4, 2010

Sheet 1 of 16

US 7,711,857 B2

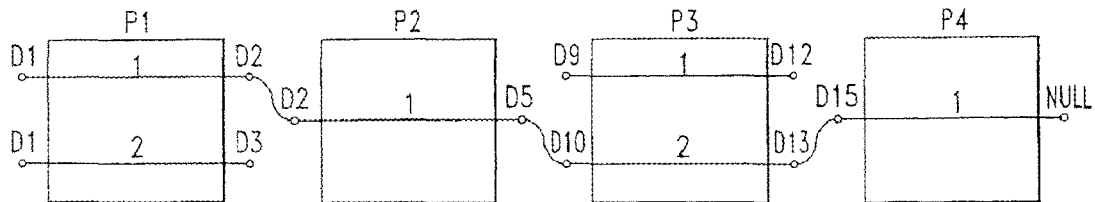
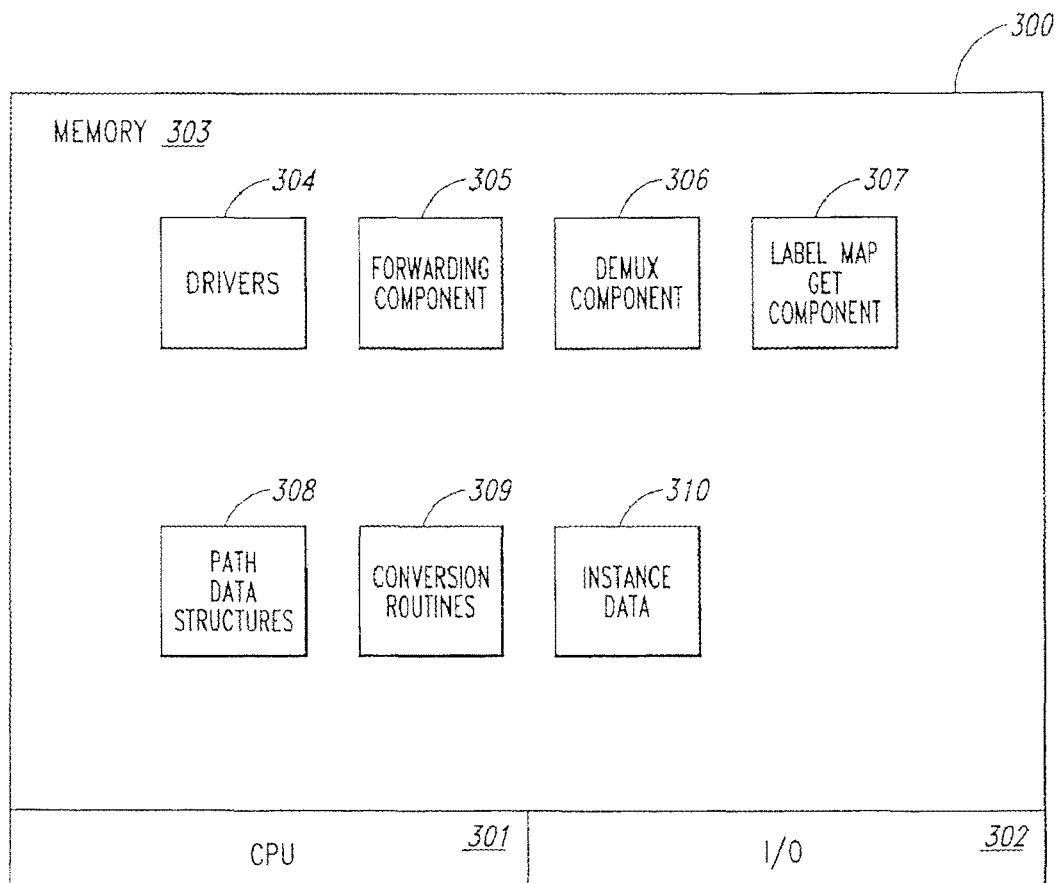
*Fig. 1*

U.S. Patent

May 4, 2010

Sheet 2 of 16

US 7,711,857 B2

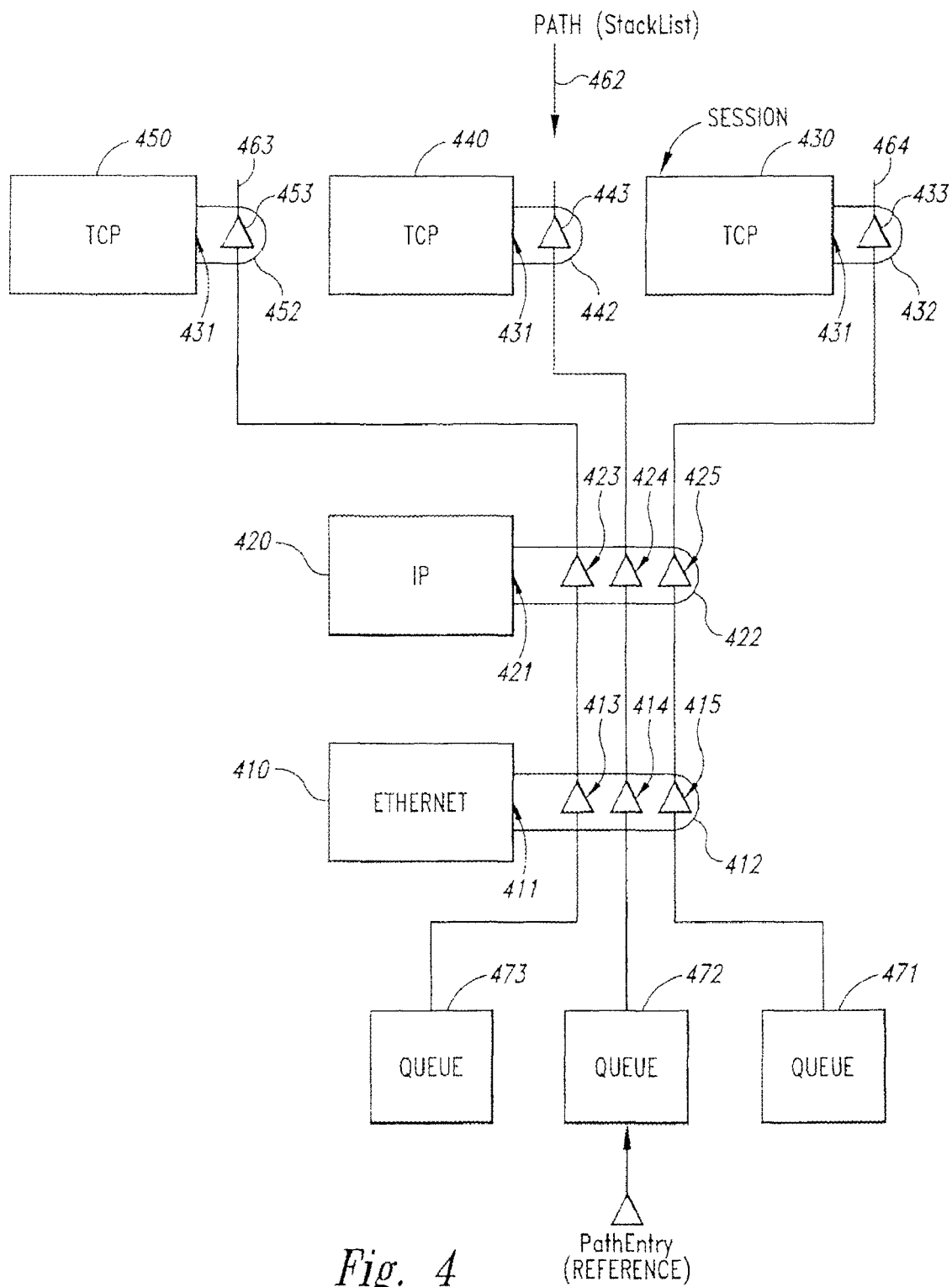
*Fig. 2**Fig. 3*

U.S. Patent

May 4, 2010

Sheet 3 of 16

US 7,711,857 B2

*Fig. 4*

U.S. Patent

May 4, 2010

Sheet 4 of 16

US 7,711,857 B2

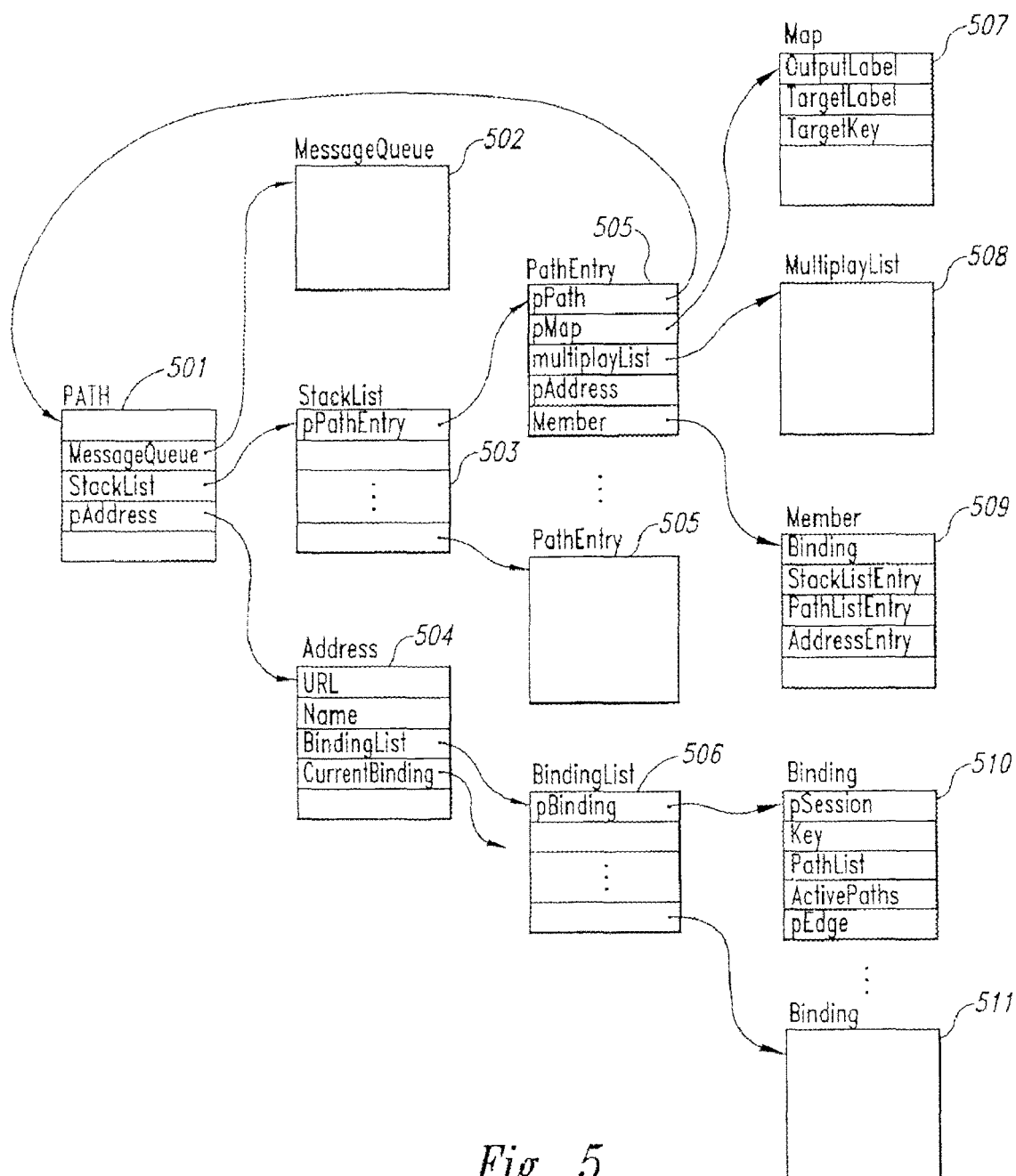
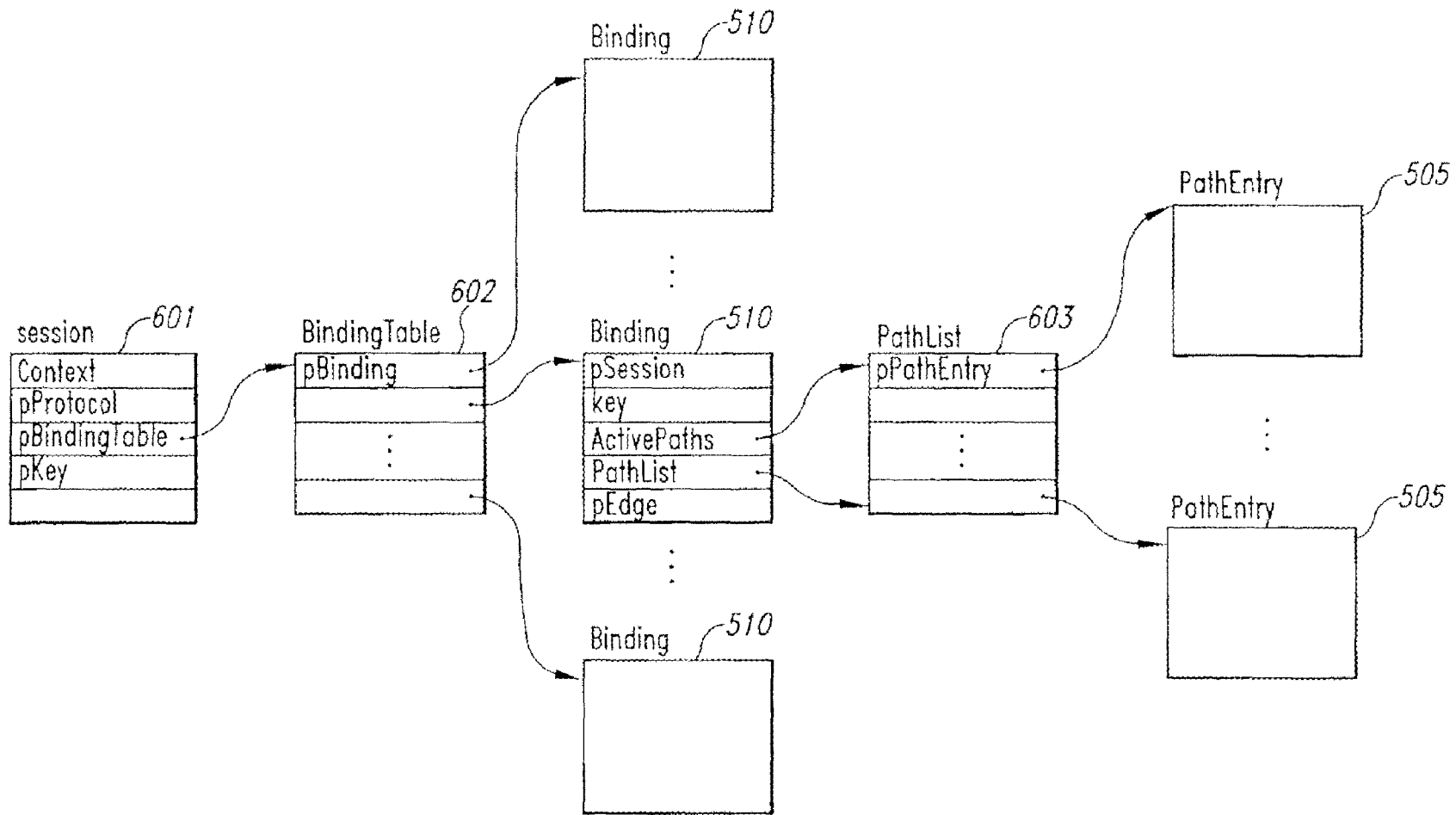


Fig. 5

*Fig. 6*

U.S. Patent

May 4, 2010

Sheet 6 of 16

US 7,711,857 B2

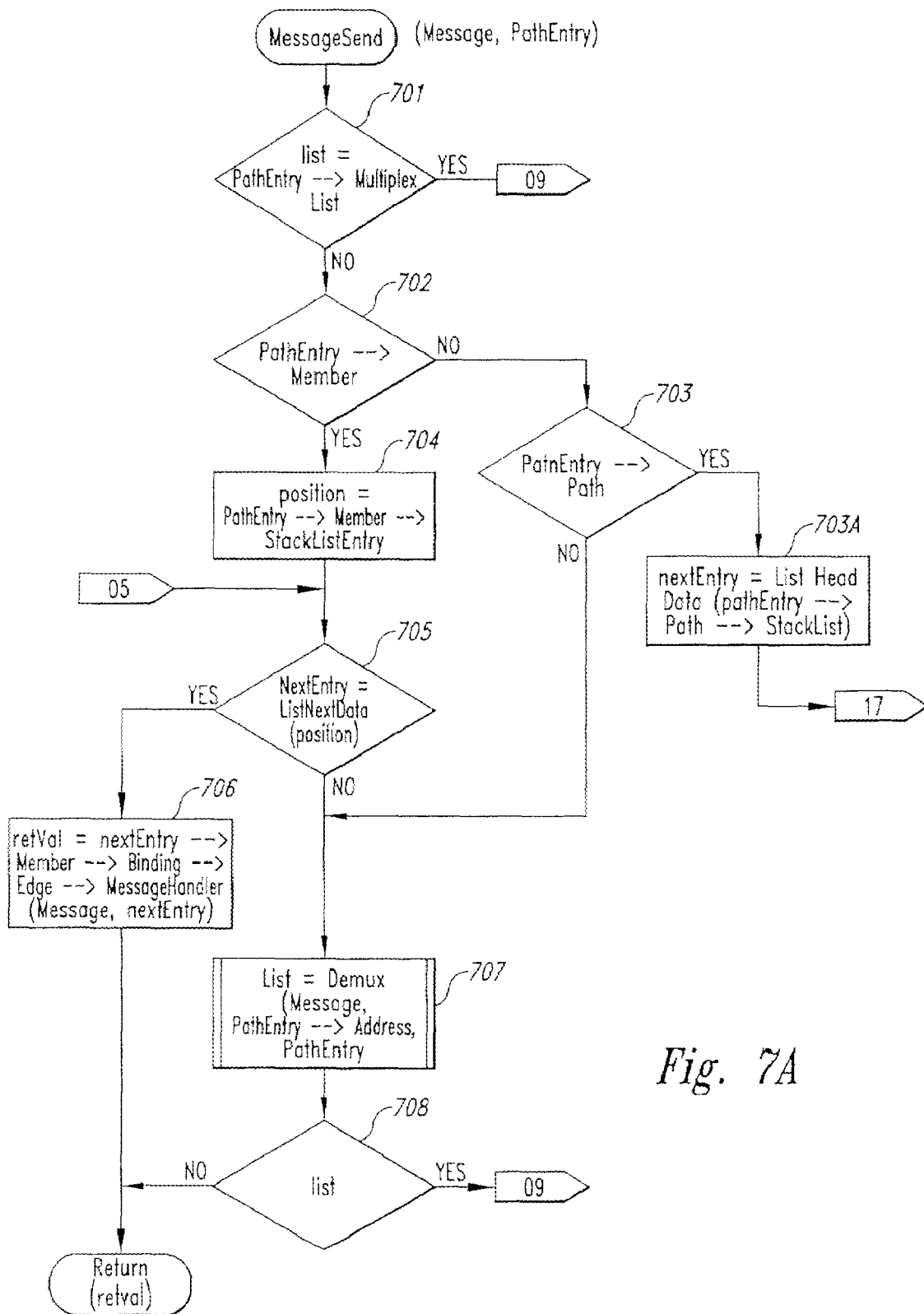


Fig. 7A

U.S. Patent

May 4, 2010

Sheet 7 of 16

US 7,711,857 B2

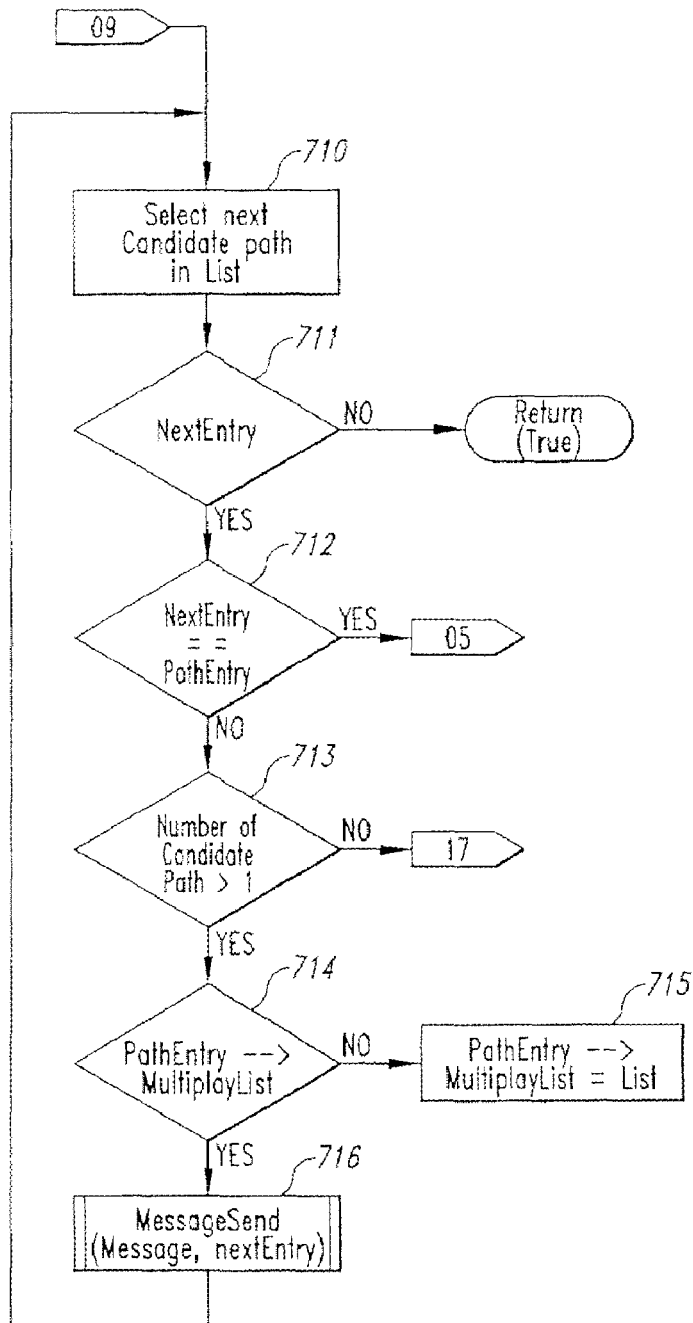


Fig. 7B

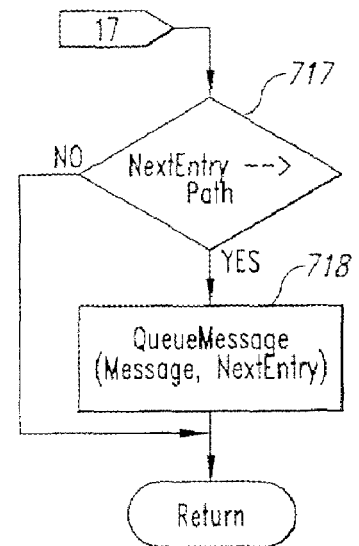


Fig. 7C

U.S. Patent

May 4, 2010

Sheet 8 of 16

US 7,711,857 B2

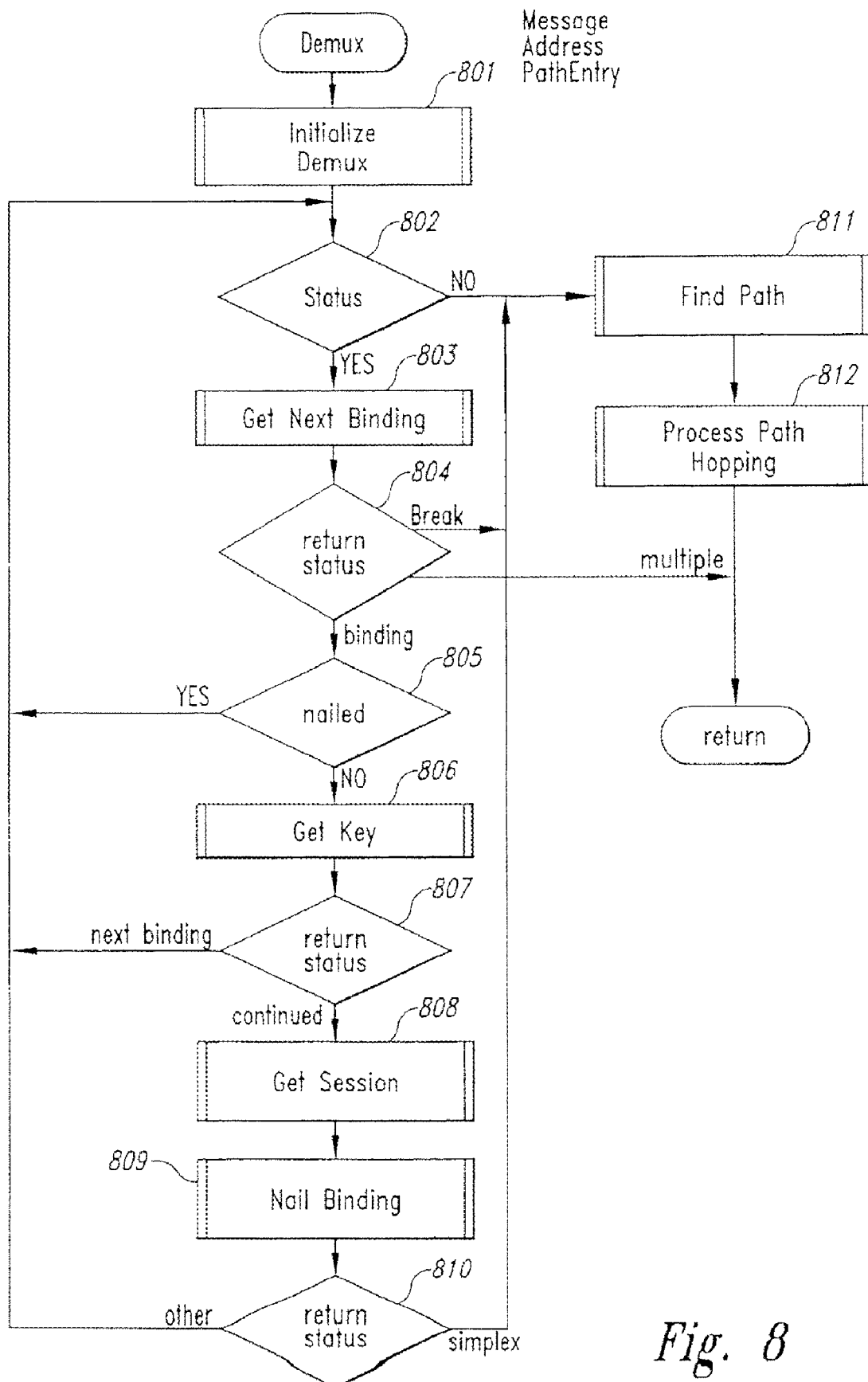


Fig. 8

U.S. Patent

May 4, 2010

Sheet 9 of 16

US 7,711,857 B2

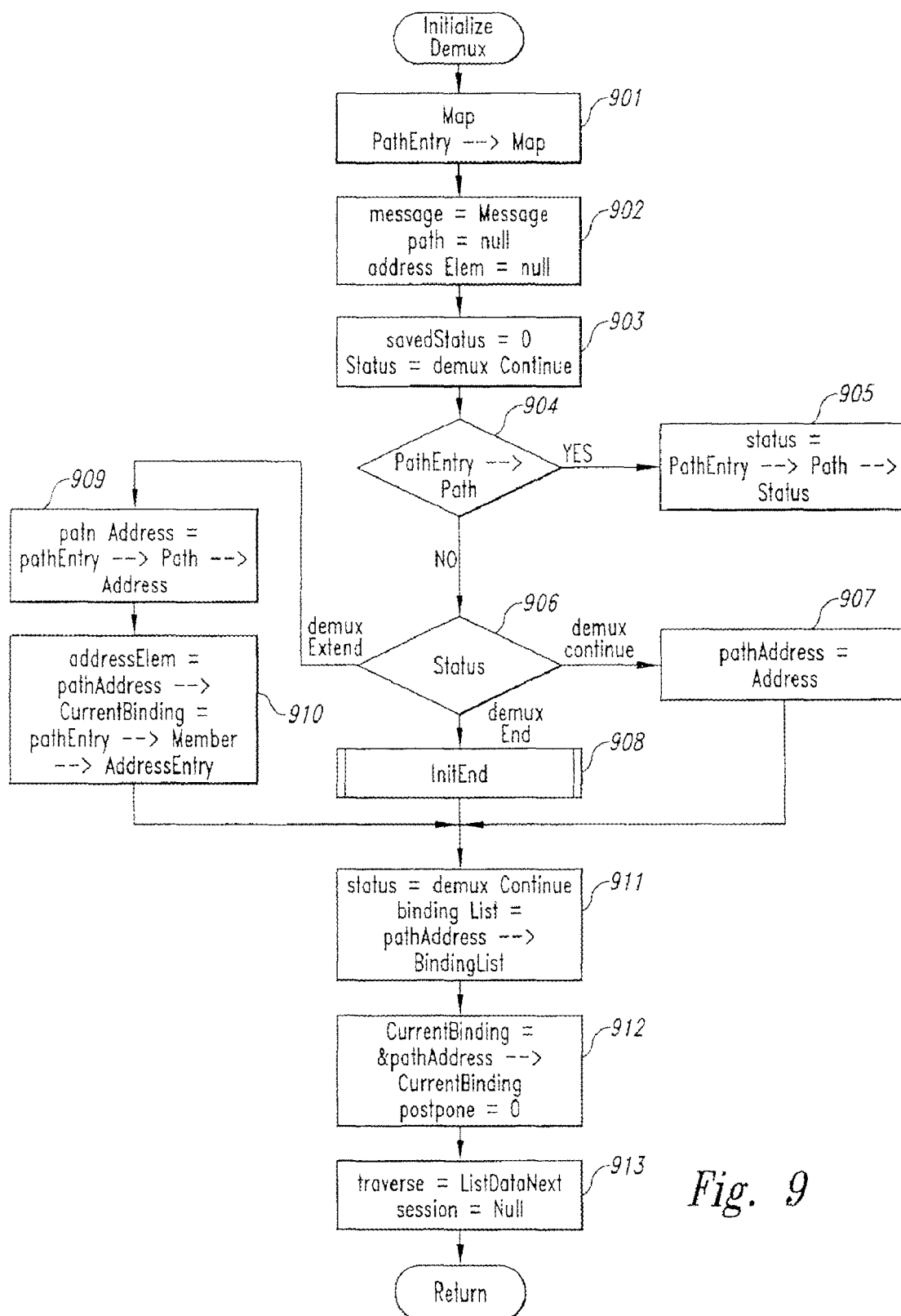


Fig. 9

U.S. Patent

May 4, 2010

Sheet 10 of 16

US 7,711,857 B2

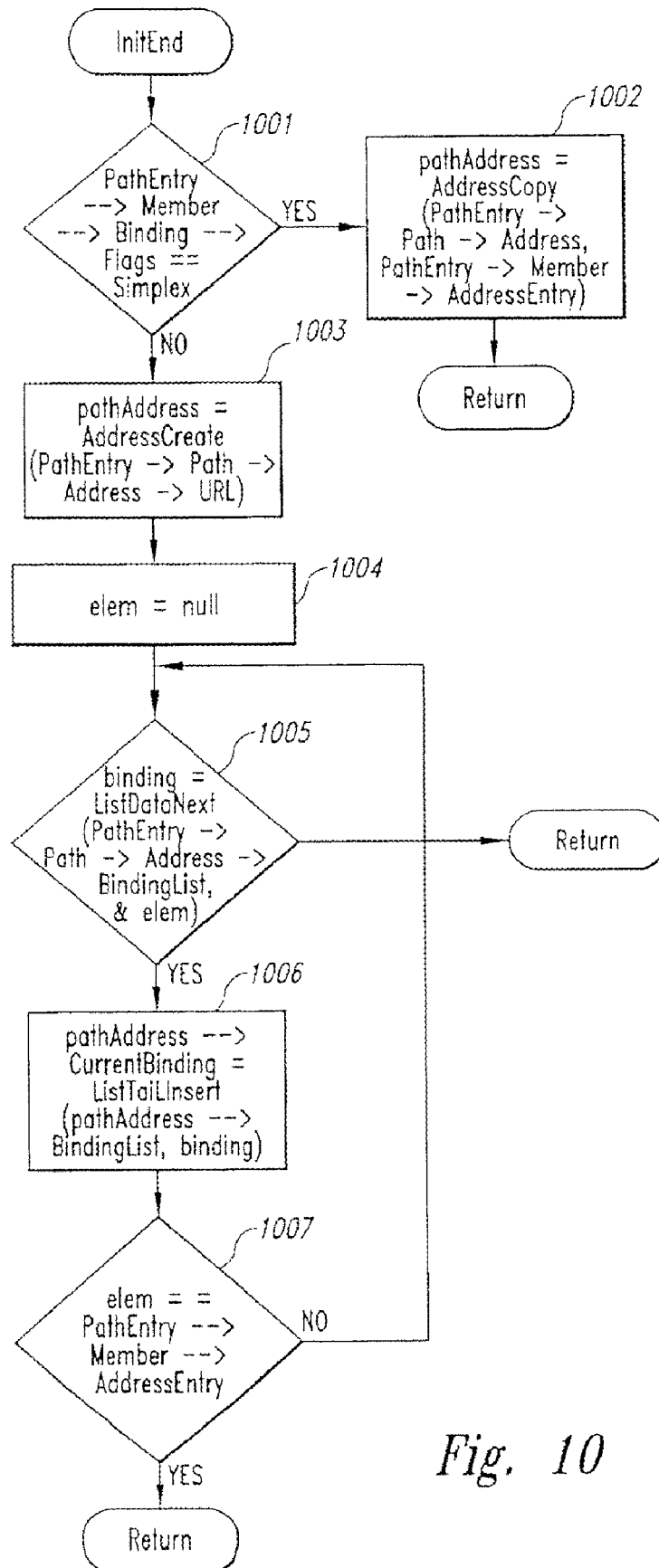


Fig. 10

U.S. Patent

May 4, 2010

Sheet 11 of 16

US 7,711,857 B2

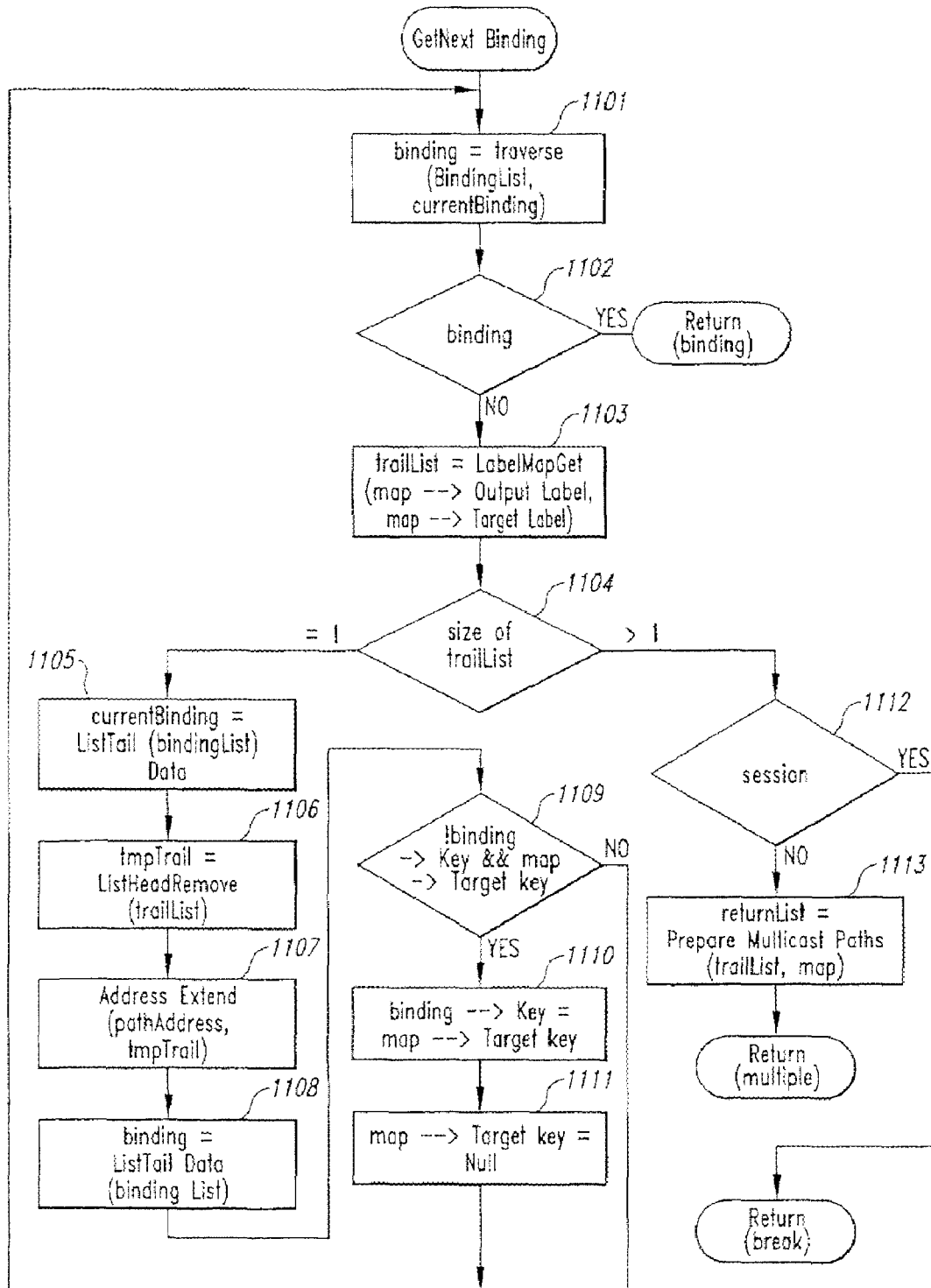


Fig. 11

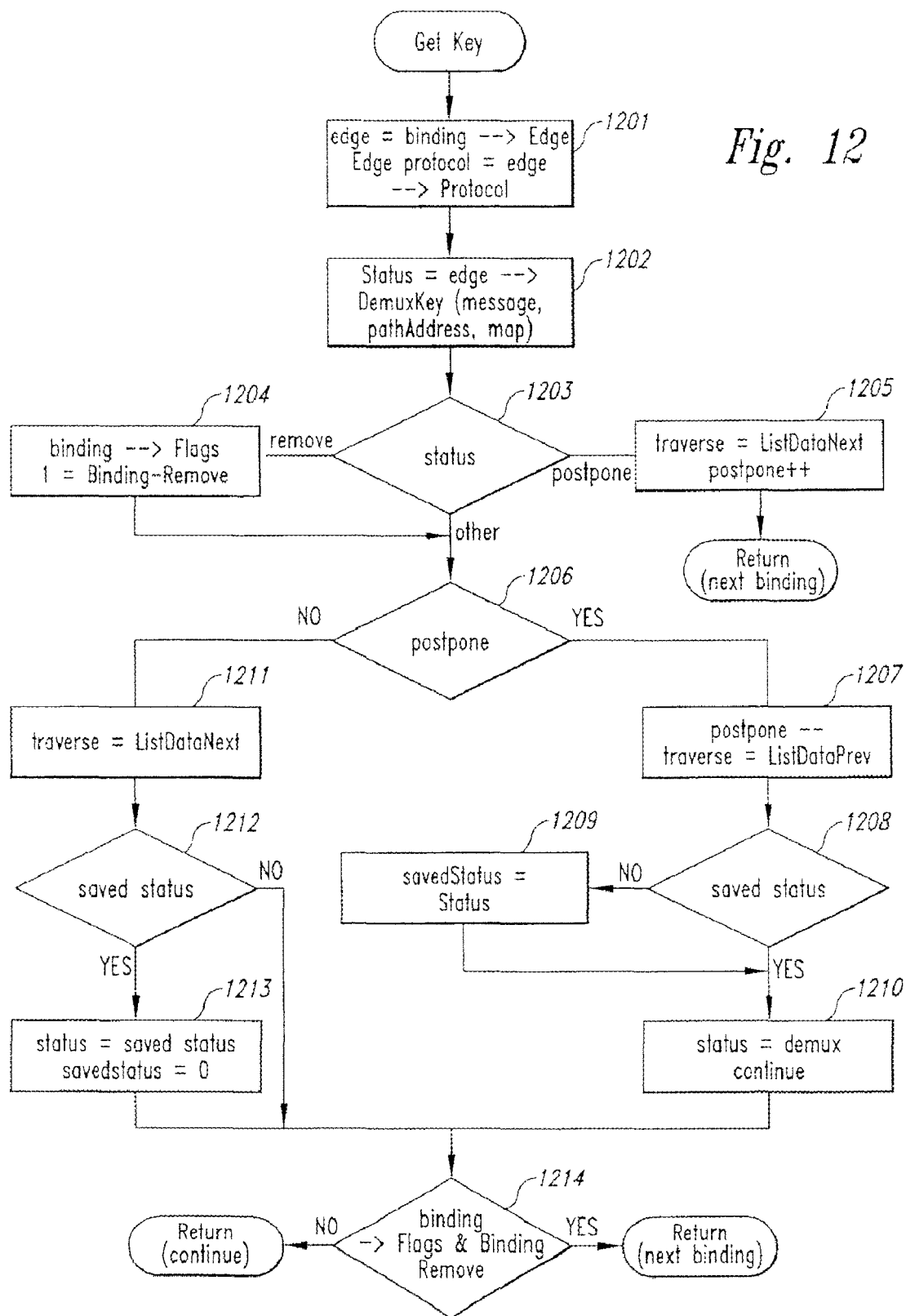
U.S. Patent

May 4, 2010

Sheet 12 of 16

US 7,711,857 B2

Fig. 12

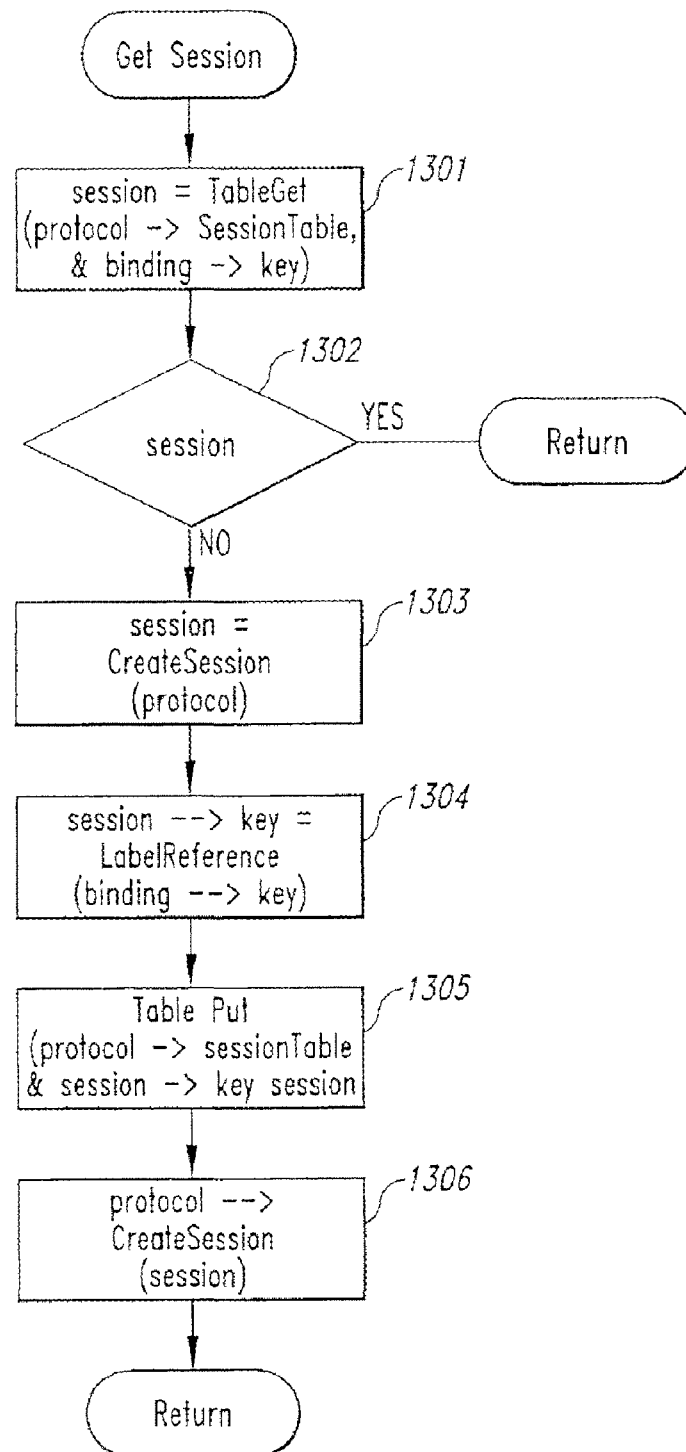


U.S. Patent

May 4, 2010

Sheet 13 of 16

US 7,711,857 B2

*Fig. 13*

U.S. Patent

May 4, 2010

Sheet 14 of 16

US 7,711,857 B2

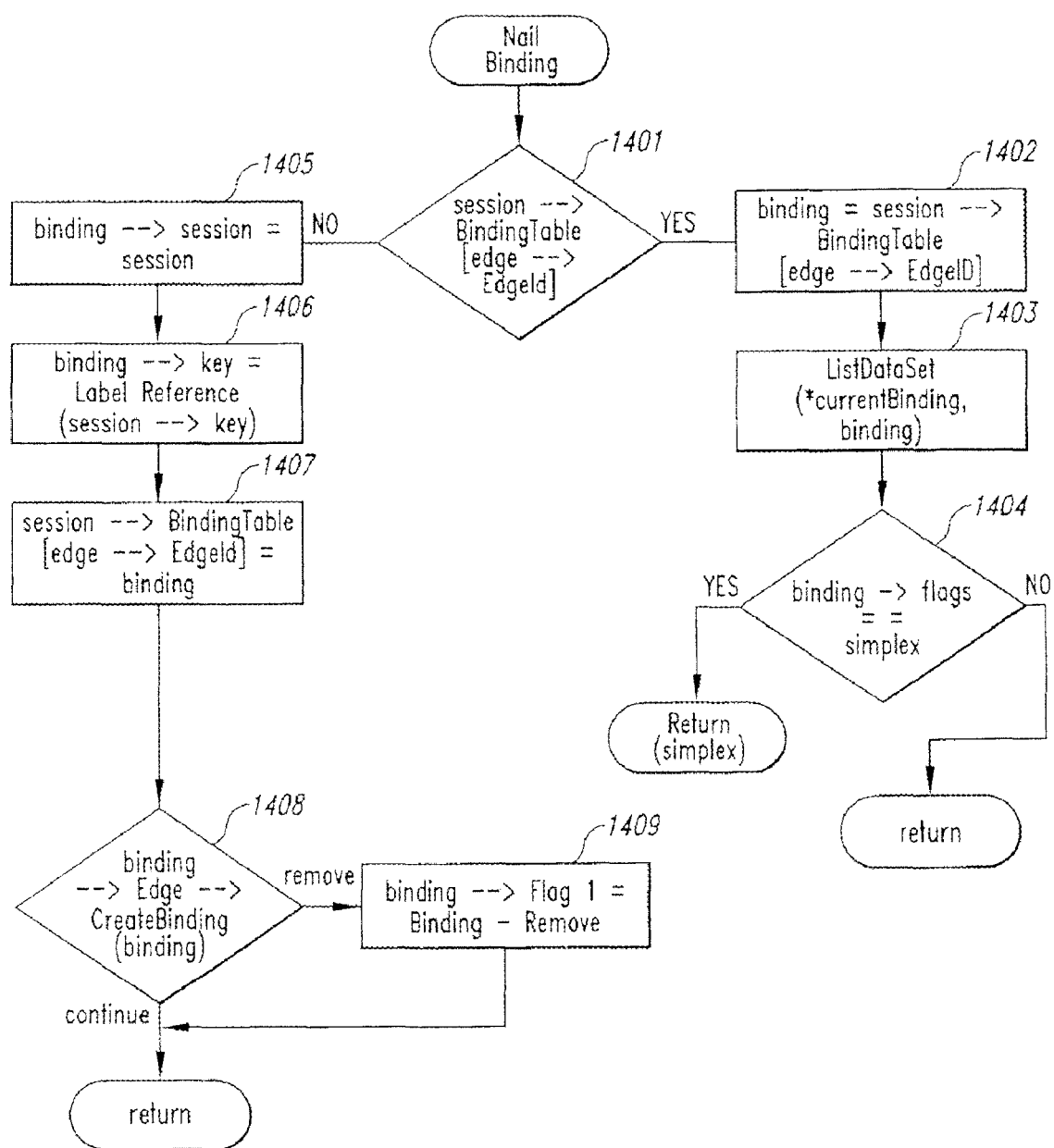


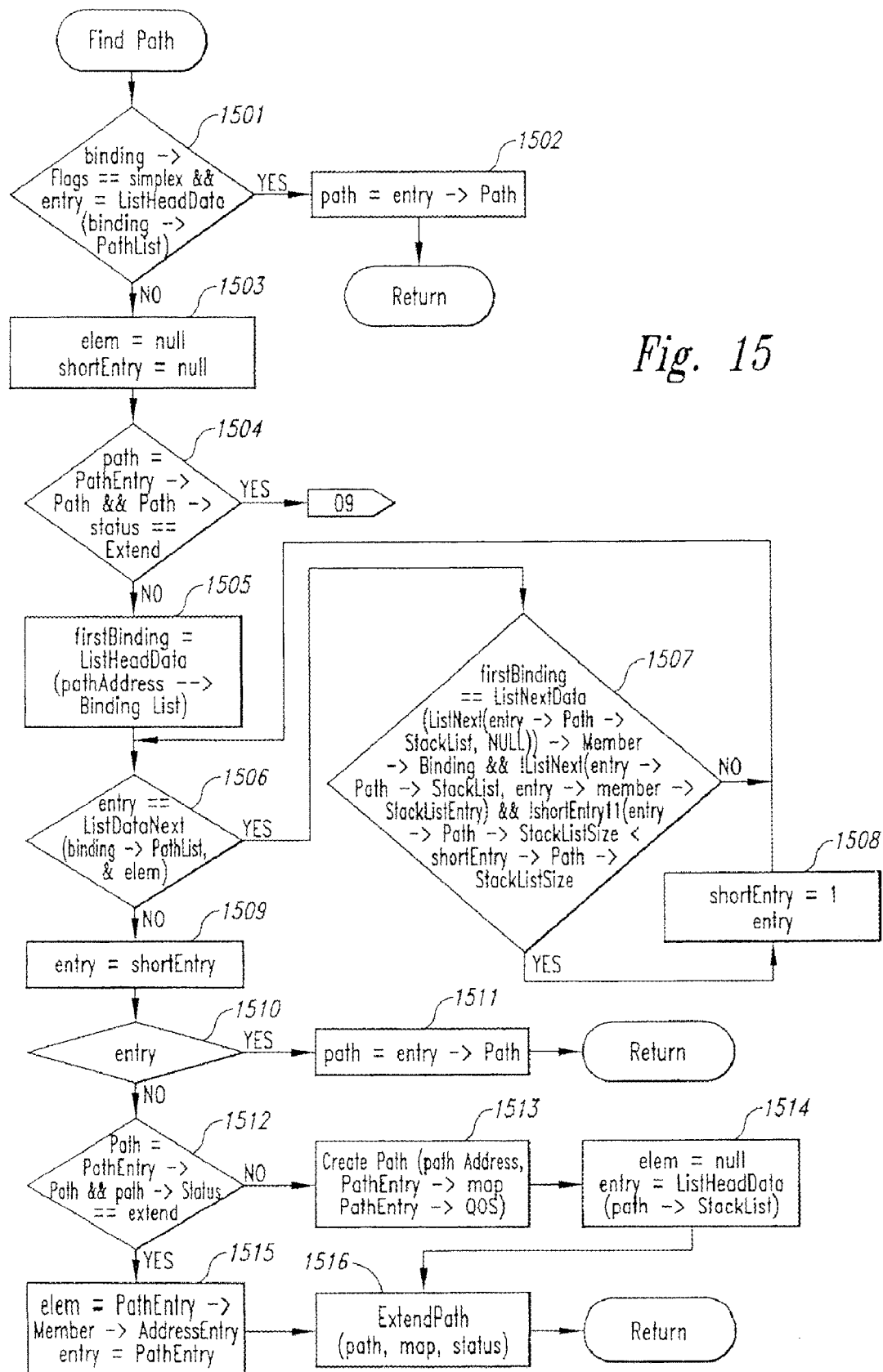
Fig. 14

U.S. Patent

May 4, 2010

Sheet 15 of 16

US 7,711,857 B2



U.S. Patent

May 4, 2010

Sheet 16 of 16

US 7,711,857 B2

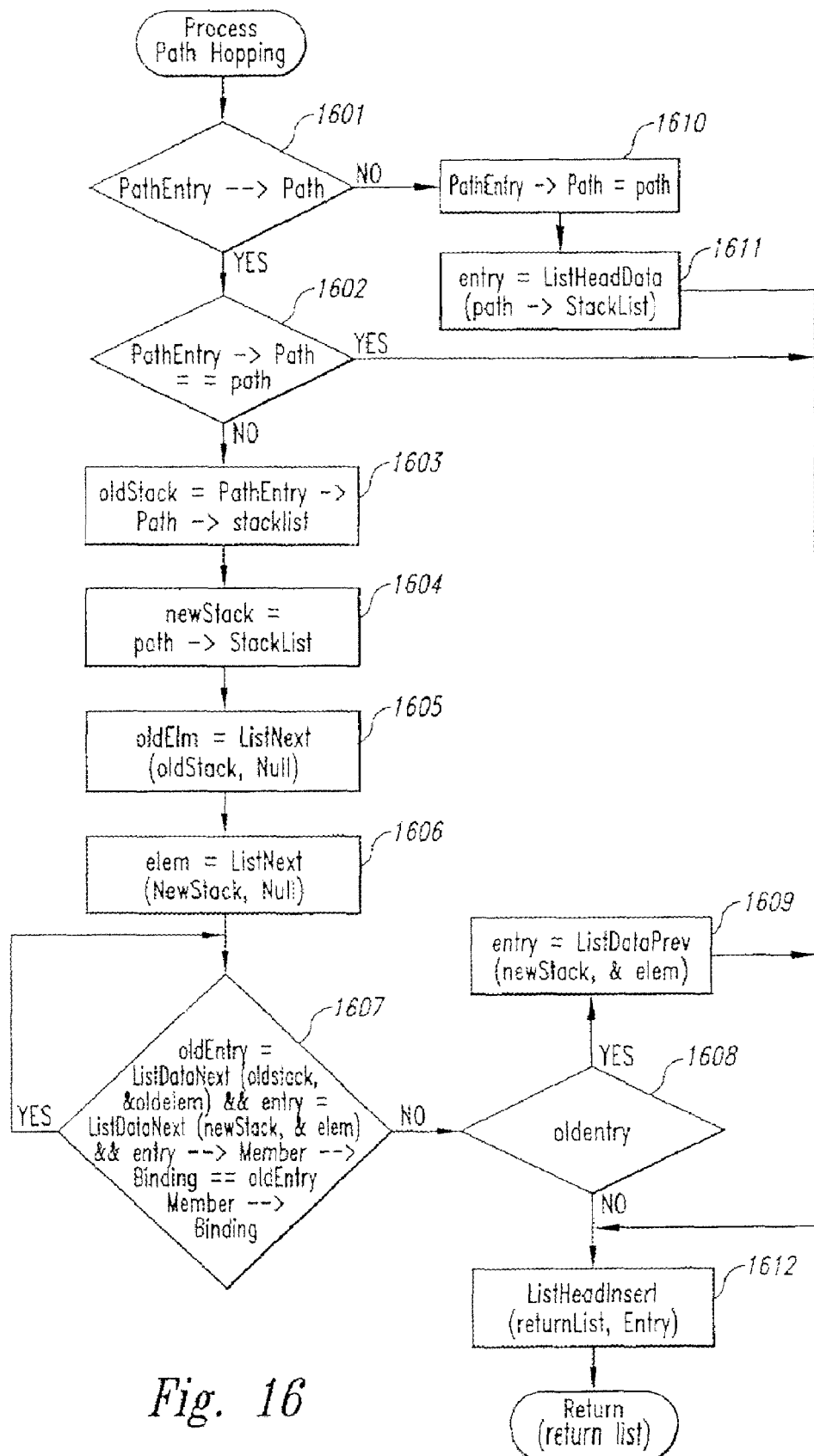


Fig. 16

US 7,711,857 B2

1

**METHOD AND SYSTEM FOR DATA
DEMULTIPLEXING****PRIORITY CLAIM**

This application is a continuation of U.S. Ser. No. 10/636,314 filed on Aug. 26, 2003 which is a continuation of Ser. No. 09/474,664, now U.S. Pat. No. 6,629,163 filed on Dec. 29, 1999.

TECHNICAL FIELD

The present invention relates generally to a computer system for data demultiplexing.

BACKGROUND

The following application is incorporated by reference as if fully set forth herein: U.S. application Ser. No. 10/636,314 filed Aug. 6, 2003.

Computer systems, which are becoming increasingly pervasive, generate data in a wide variety of formats. The Internet is an example of interconnected computer systems that generate data in many different formats. Indeed, when data is generated on one computer system and is transmitted to another computer system to be displayed, the data may be converted in many different intermediate formats before it is eventually displayed. For example, the generating computer system may initially store the data in a bitmap format. To send the data to another computer system, the computer system may first compress the bitmap data and then encrypt the compressed data. The computer system may then convert that compressed data into a TCP format and then into an IP format. The IP formatted data may be converted into a transmission format, such as an ethernet format. The data in the transmission format is then sent to a receiving computer system. The receiving computer system would need to perform each of these conversions in reverse order to convert the data in the bitmap format. In addition, the receiving computer system may need to convert the bitmap data into a format that is appropriate for rendering on output device.

In order to process data in such a wide variety of formats, both sending and receiving computer systems need to have many conversion routines available to support the various formats. These computer systems typically use predefined configuration information to load the correct combination of conversion routines for processing data. These computer systems also use a process-oriented approach when processing data with these conversion routines. When using a process-oriented approach, a computer system may create a separate process for each conversion that needs to take place. A computer system in certain situations, however, can be expected to receive data and to provide data in many different formats that may not be known until the data is received. The overhead of statically providing each possible series of conversion routines is very high. For example, a computer system that seizes as a central controller for data received within a home would be expected to process data received via telephone lines, cable TV lines, and satellite connections in many different formats. The central controller would be expected to output the data to computer displays, television displays, entertainment centers, to speakers, recording devices, and so on in many different formats. Moreover, since the various conversion routines may be developed by different organizations, it may not be easy to identify that the output format of one conversion routine is compatible with the input format of another conversion routine.

2

It would be desirable to have a technique for dynamically identifying a series of conversion routines for processing data. In addition, it would be desirable to have a technique in which the output format of one conversion routine can be identified as being compatible with the input format of another conversion routine. It would also be desirable to store the identification of a series of conversion routines so that the series can be quickly identified when data is received.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating example processing of a message by the conversion system.

FIG. 2 is a block diagram illustrating a sequence of edges.

FIG. 3 is a block diagram illustrating components of the conversion system in one embodiment.

FIG. 4 is a block diagram illustrating example path data structures in one embodiment.

FIG. 5 is a block diagram that illustrates the interrelationship of the data structures of a path.

FIG. 6 is a block diagram that illustrates the interrelationship of the data structures associated with a session.

FIGS. 7A, 7B, and 7C comprise a flow diagram illustrating the processing of the message send routine.

FIG. 8 is a flow diagram of the demux routine.

FIG. 9 is a flow diagram of the initialize demux routine.

FIG. 10 is a flow diagram of the init end routine.

FIG. 11 is a flow diagram of a routine to get the next binding.

FIG. 12 is a flow diagram of the get key routine.

FIG. 13 is a flow diagram of the get session routine.

FIG. 14 is a flow diagram of the nail binding routine.

FIG. 15 is a flow diagram of the find path routine.

FIG. 16 is a flow diagram of the process of path hopping routine.

DETAILED DESCRIPTION

A method and system for converting a message that may contain multiple packets from a source format into a target format. When a packet of a message is received, the conversion system in one embodiment searches for and identifies a sequence of conversion routines (or more generally message handlers) for processing the packets of the message by comparing the input and output formats of the conversion routines. (A message is a collection of data that is related in some way, such as stream of video or audio data or an email message.) The identified sequence of conversion routines is used to convert the message from the source format to the target format using various intermediate formats. The conversion system then queues the packet for processing by the identified sequence of conversion routines. The conversion system stores the identified sequence so that the sequence can be quickly found (without searching) when the next packet in the message is received. When subsequent packets of the message are received, the conversion system identifies the sequence and queues the packets for no pressing by the sequence. Because the conversion system receives multiple messages with different source and target formats and identifies a sequence of conversion routines for each message, the conversion systems effectively "demultiplexes" the messages. That is, the conversion system demultiplexes the messages by receiving the message, identifying the sequence of conversion routines, and controlling the processing of each message by the identified sequence. Moreover, since the conversion routines may need to retain state information between the receipt of one packet of a message and the next packet of

US 7,711,857 B2

3

that message, the conversion system maintains state information as an instance or session of the conversion routine. The conversion system routes all packets for a message through the same session of each conversion routine so that the same state or instance information can be used by all packets of the message. A sequence of sessions of conversion routines is referred to as a "path." In one embodiment, each path has a path thread associated with it for processing of each packet destined for that path.

In one embodiment, the packets of the messages are initially received by "drivers," such as an Ethernet driver. When a driver receives a packet, it forwards the packet to a forwarding component of the conversion system. The forwarding component is responsible for identifying the session of the conversion routine that should next process the packet and invoking that conversion routine. When invoked by a driver, the forwarding component may use a demultiplexing ("demux") component to identify the session of the first conversion routine of the path that is to process the packet and then queues the packet for processing by the path. A path thread is associated with each path. Each path thread is responsible for retrieving packets from the queue of its path and forwarding the packets to the forwarding component. When the forwarding component is invoked by a path thread, it initially invokes the first conversion routine in the path. That conversion routine processes the packet and forwards the processed packet to the forwarding component, which then invokes the second conversion routine in the path. The process of invoking the conversion routines and forwarding the processed packet to the next conversion routine continues until the last conversion routine in the path is invoked. A conversion routine may defer invocation of the forwarding component until it aggregates multiple packets or may invoke the forwarding component multiple times for a packet once for each sub-packet.

The forwarding component identifies the next conversion routine in the path using the demux component and stores that identification so that the forwarding component can quickly identify the conversion routine when subsequent packets of the same message are received. The demux component searches for the conversion routine and session that is to next process a packet. The demux component then stores the identification of the session and conversion routine as part of a path data structure so that the conversion system does not need to search for the session and conversion routine when requested to demultiplex subsequent packets of the same message. When searching for the next conversion routine, the demux component invokes a label map get component that identifies the next conversion routine. Once the conversion routine is found, the demux component identifies the session associated with that message by, in one embodiment, invoking code associated with the conversion routine. In general, the code of the conversion routine determines what session should be associated with a message. In certain situations, multiple messages may share the same session. The demux component then extends the path for processing that packet to include that session and conversion routine. The sessions are identified so that each packet is associated with the appropriate state information. The dynamic identification of conversion routines is described in U.S. patent application Ser. No. 09/304,973, filed on May 4, 1999, entitled "Method and System for Generating a Mapping Between Types of Data," which is hereby incorporated by reference.

FIG. 1 is a block diagram illustrating example processing of a message by the conversion system. The driver 101 receives the packets of the message from a network. The driver performs any appropriate processing of the packet and invokes a message send routine passing the processed packet

4

along with a reference path entry 150. The message send routine is an embodiment of the forwarding component. A path is represented by a series of path entries, which are represented by triangles. Each member path entry represents a session and conversion routine of the path, and a reference path entry represents the overall path. The passed reference path entry 150 indicates to the message send routine that it is being invoked by a driver. The message send routine invokes the demux routine 102 to search for and identify the path of sessions that is to process the packet. The demux routine may in turn invoke the label map get routine 104 to identify a sequence of conversion routines for processing the packet. In this example, the label map get routine identifies the first three conversion routines, and the demux routine creates the member path entries 151, 152, 153 of the path for these conversion routines. Each path entry identifies a session for a conversion routine, and the sequence of path entries 151-155 identifies a path. The message send routine then queues the packet on the queue 149 for the path that is to process the packets of the message. The path thread 105 for the path retrieves the packet from the queue and invokes the message send routine 106 passing the packet and an indication of the path. The message send routine determines that the next session and conversion routine as indicated by path entry 151 has already been found. The message send routine then invokes the instance of the conversion routine for the session. The conversion routine processes the packet and then invokes the message send routine 107. This processing continues until the message send routine invokes the demux routine 110 after the packet is processed by the conversion routine represented by path entry 153. The demux routine examines the path and determines that it has no more path entries. The demux routine then invokes the label map get routine 111 to identify the conversion routines for further processing of the packet. When the conversion routines are identified, the demux routine adds path entries 154, 155 to the path. The message send routine invokes the conversion routine associated with path entry 154. Eventually, the conversion routine associated with path entry 155 performs the final processing for the path.

The label map get routine identifies a sequence of "edges" for converting data in one format into another format. Each edge corresponds to a conversion routine for converting data from one format to another. Each edge is part of a "protocol" (or more generally a component) that may include multiple related edges. For example, a protocol may have edges that each convert data in one format into several different formats. Each edge has an input format and an output format. The label map get routine identifies a sequence of edges such that the output format of each edge is compatible with the input format of another edge in the sequence, except for the input format of the first edge in the sequence and the output format of the last edge in the sequence. FIG. 2 is a block diagram illustrating a sequence of edges. Protocol P1 includes an edge for converting format D1 to format D2 and an edge for converting format D1 to format D3; protocol P2 includes an edge for converting format D2 to format D5, and so on. A sequence for converting format D1 to format D15 is shown by the curved lines and is defined by the address "P1:1, P2:1, P3:2, P4:7." When a packet of data, in format D1 is processed by this sequence, it is converted to format D15. During the process the packet of data is sequentially converted to format D2, D5, and D13. The output format of protocol P2, edge 1 (i.e., P2:1) is format D5, but the input format of P3:2 is format D10. The label map get routine uses an aliasing mechanism by which two formats, such as D5 and D10 are identified as being compatible. The use of aliasing allows different names of the same format or compatible formats to be correlated.

US 7,711,857 B2

5

FIG. 3 is a block diagram illustrating components of the conversion system in one embodiment. The conversion system 300 can operate on a computer system with a central processing unit 301, I/O devices 302, and memory 303. The I/O devices may include an Internet connection, a connection to various output devices such as a television, and a connection to various input devices such as a television receiver. The media mapping system may be stored as instructions on a computer-readable medium, such as a disk drive, memory, or data transmission medium. The data structures of the media mapping system may also be stored on a computer-readable medium. The conversion system includes drivers 304, a forwarding component 305, a demux component 306, a label map get component 307, path data structures 308, conversion routines 309, and instance data 310. Each driver receives data in a source format and forwards the data to the forwarding component. The forwarding component identifies the next conversion routine in the path and invokes that conversion routine to process a packet. The forwarding component may invoke the demux component to search for the next conversion routine and add that conversion routine to the path. The demux component may invoke the label map get component to identify the next conversion routine to process the packet. The demux component stores information defining the paths in the path structures. The conversion routines store their state information in the instance data.

FIG. 4 is a block diagram illustrating example path data structures in one embodiment. The demux component identifies a sequence of “edges” for converting data in one format into another format by invoking the label map get component. Each edge corresponds to a conversion routine for converting data from one format to another. As discussed above, each edge is part of “protocol” that may include multiple related so edges. For example, a protocol may have edges that each convert data in one format into several different formats. Each edge has as an input format (“input label”) and an output format (“output label”). Each rectangle represents a session 410, 420, 430, 440, 450 for a protocol. A session corresponds to an instance of a protocol. That is, the session includes the protocol and state information associated with that instance of the protocol. Session 410 corresponds to a session for an Ethernet protocol; session 420 corresponds to a session for an IP protocol; and sessions 430, 440, 450 correspond to sessions for a TCP protocol. FIG. 4 illustrates three paths 461, 462, 463. Each path includes edges 417, 421, 431. The paths share the same Ethernet session 410 and IP session 420, but each path has a unique TCP session 430, 440, 450. Thus, path 461 includes sessions 410, 420, and 430; path 462 includes sessions 410, 420, and 440; and path 463 includes sessions 410, 420, and 450. The conversion system represents each path by a sequence of path entry structures. Each path entry structure is represented by a triangle. Thus, path 461 is represented by path entries 415, 425, and 433. The conversion system represents the path entries of a path by a stack list. Each path also has a queue 471, 472, 473 associated with it. Each queue stores the messages that are to be processed by the conversion routines of the edges of the path. Each session includes a binding (412, 422, 432, 442, 452) that is represented by an oblong shape adjacent to the corresponding edge. A binding for an edge of a session represents those paths that include the edge. The binding 412 indicates that three paths are bound (or “nailed”) to edge 411 of the Ethernet session 410. The conversion system uses a path list to track the paths that are bound to a binding. The path list of binding 412 identifies path entries 413, 414, and 415.

FIG. 5 is a block diagram that illustrates the interrelationship of the data structures of a path. Each path has a corre-

6

sponding path structure 501 that contains status information and pointers to a message queue structure 502, a stack list structure 503, and a path address structure 504. The status of a path can be extend, continue, or end. Each message handler returns a status for the path. The status of extend means that additional path entries should be added to the path. The status of end means that this path should end at this point and subsequent processing should continue at a new path. The status of continue means that the protocol does not care how the path is handled. In one embodiment, when a path has a status of continue, the system creates a copy of the path and extends the copy. The message queue structure identifies the messages (or packets of a message) that are queued up for processing by the path and identifies the path entry at where the processing should start. The stack list structure contains a list of pointers to the path entry structures 505 that comprise the path. Each path entry structure contains a pointer to the corresponding path data structure, a pointer to a map structure 507, a pointer to a multiplex list 508, a pointer to the corresponding path address structure, and a pointer to a member structure 509. A map structure identifies the output label of the edge of the path entry and optionally a target label and a target key. A target key identifies the sessions associated with the protocol that converts the packet to the target label. (The terms “media,” “label,” and “format” are used interchangeably to refer to the output of a protocol.) The multiplex list is used during the demux process to track possible next edges when a path is being identified as having more than one next edge. The member structure indicates that the path entry represents an edge of a path and contains a pointer to a binding structure to which the path entry is associated (or “nailed”), a stack list entry is the position of the path entry within the associated stack list, a path list entry is the position of the path entry within the associated path list of a binding and an address entry is the position of the binding within the associated path address. A path address of a path identifies the bindings to which the path entries are bound. The path address structure contains a URL for the path, the name of the path identified by the address, a pointer to a binding list structure 506, and the identification of the current binding within the binding list. The URL (e.g., “protocol://tcp(0)/ip(0)/eth(0)”) identifies conversion routines (e.g. protocols and edges) of a path in a human-readable format. The URL (universal resource locator) includes a type field (e.g., “protocol”) followed by a sequence of items (e.g., “tcp(0)”). The type field specifies the format of the following information in the URL, that specifies that the type field is followed by a sequence of items. Each item identifies a protocol and an edge (e.g., the protocol is “tcp” and the edge is “0”). In one embodiment, the items of a URL may also contain an identifier of state information that is to be used when processing a message. These URLs can be used to illustrate to a user various paths that are available for processing a message. The current binding is the last binding in the path as the path is being built. The binding list structure contains a list of pointers to the binding structures associated with the path. Each binding structure 510 contains a pointer to a session structure, a pointer to an edge structure, a key, a path list structure, and a list of active paths through the binding. The key identifies the state information for a session of a protocol. A path list structure contains pointers to the path entry structures associated with the binding.

FIG. 6 is a block diagram that illustrates the interrelationship of the data structures associated with a session. A session structure 601 contains the context for the session, a pointer to a protocol structure for the session, a pointer to a binding table structure 602 for the bindings associated with the session, and

US 7,711,857 B2

7

the key. The binding table structure contains a list of pointers to the binding structures **510** for the session. The binding structure is described above with reference to FIG. **5**. The path list structure **603** of the binding structure contains a list of pointers to path entry structures **505**. The path entry structures are described with reference to FIG. **5**.

FIGS. **7A**, **7B**, and **7C** comprise a flow diagram illustrating the processing of the message send routine. The message send routine is passed a message along with the path entry associated with the session that last processed the message. The message send routine invokes the message handler of the next edge in the path or queues the message for processing by a path. The message handler invokes the demux routine to identify the next path entry of the path. When a driver receives a message, it involves the message send routine passing a reference path entry. The message send routine examines the passed path entry to determine (1) whether multiple paths branch from the path of the passed path entry, (2) whether the passed path entry is a reference with an associated path, or (3) whether the passed path entry is a member with a next path entry. If multiple paths branch from the path of the passed path entry, then the routine recursively invokes the message send routine for each path. If the path entry is a reference with an associated path, then the driver previously invoked the message send routine, which associated a path with the reference path entry, and the routine places the message on the queue for the path. If the passed path entry is a member with a next path entry, then the routine invokes the message handler (i.e., conversion routine of the edge) associated with the next path entry. If the passed path entry is a reference without an associated path or is a member without a next path entry, then the routine invokes the demux routine to identify the next path entry. The routine then recursively invokes the message send routine passing that next path entry. In decision block **701**, if the passed path entry has a multiplex list, then the path branches off into multiple paths and the routine continues at block **709**, else the routine continues at block **702**. A packet may be processed by several different paths. For example, if a certain message is directed to two different output devices, then the message is processed by two different paths. Also, a message may need to be processed by multiple partial paths when searching for a complete path. In decision block **702**, if the passed path entry is a member, then either the next path entry indicates a nailed binding or the path needs to be extended and the routine continues at block **704**, else the routine continues at block **703**. A nailed binding is a binding (e.g., edge and protocol) is associated with a session. In decision block **703**, the passed path entry is a reference and if the passed path entry has an associated path, then the routine can queue the message for the associated path and the routine continues at block **703A**, else the routine needs to identify a path and the routine continues at block **707**. In block **703A**, the routine sets the entry to the first path entry in the path and continues at block **717**. In block **704**, the routine sets the variable position to the stack list entry of the passed path entry. In decision block **705**, the routine sets the variable next entry to the next path entry in the path. If there is a next entry in the path, then the next session and edge of the protocol have been identified and the routine continues at block **706**, else the routine continues at block **707**. In block **706**, the routine passes the message to the message handler of the edge associated with the next entry and then returns. In block **706**, the routine invokes the demux routine passing the passed message, the address of the passed path entry, and the passed path entry. The demux routine returns a list of candidate paths for processing of the message. In decision block **708**, if at least

8

one candidate path is returned, then the routine continues at block **709**, else the routine returns.

Blocks **709-716** illustrate the processing of a list of candidate paths that extend from the passed path entry. In blocks **710-716**, the routine loops selecting each candidate path and sending the message to be process by each candidate path. In block **710**, the routine sets the next entry to the first path entry of the next candidate path. In decision block **711**, if all the candidate paths have not yet been processed, then the routine continues at block **712**, else the routine returns. In decision block **712**, if the next entry is equal to the passed path entry, then the path is to be extended and the routine continues at block **705**, else the routine continues at block **713**. The candidate paths include a first path entry that is a reference path entry for new paths or that is the last path entry of a path being extended. In decision block **713**, if the number of candidate paths is greater than one, then the routine continues at block **714**, else the routine continues at block **718**. In decision block **714**, if the passed path entry has a multiplex list associated with it, then the routine continues at block **716**, else the routine continues at block **715**. In block **715**, the routine associates the list of candidate path with the multiplex list of the passed path entry and continues at block **716**. In block **716**, the routine sends the message to the next entry by recursively invoking the message send routine. The routine then loops to block **710** to select the next entry associated with the next candidate path.

Blocks **717-718** are performed when the passed path entry is a reference path entry that has a path associated with it. In block **717**, if there is a path associated with the next entry, then the routine continues at block **718**, else the routine returns. In block **718**, the routine queues the message for the path of the next entry and then returns.

FIG. **8** is a flow diagram of the demux routine. This routine is passed the packet (message) that is received, an address structure, and a path entry structure. The demux routine extends a path, creating one if necessary. The routine loops identifying the next binding (edge and protocol) that is to process the message and "nailing" the binding, to a session for the message, if not already nailed. After identifying the nailed binding, the routine searches for the shortest path through the nailed binding, creating a path if none exists. In block **801**, the routine invokes the initialize demux routine. In blocks **802-810**, the routine loops identifying a path or portion of a path for processing the passed message. In decision block **802**, if there is a current status, which was returned by the demux routine that was last invoked (e.g., continue, extend, end, or postpone), then the routine continues at block **803**, else the routine continues at block **811**. In block **803**, the routine invokes the get next binding routine. The get next binding routine returns the next binding in the path. The binding is the edge of a protocol. That routine extends the path as appropriate to include the binding. The routine returns a return status of break, binding, or multiple. The return status of binding indicates that the next binding in the path was found by extending the path as appropriate and the routine continues to "nail" the binding to a session as appropriate. The return status of multiple means that multiple trails (e.g., candidate paths) were identified as possible extensions of the path. In a decision block **804**, if the return status is break, then the routine continues at block **811**. If the return status is multiple, then the routine returns. If the return status is binding, then the routine continues at block **805**. In decision block **805**, if the retrieved binding is nailed as indicated by being assigned to a session, then the routine loops to block **802**, else the routine continues at block **806**. In block **806**, the routine invokes the get key routine of the edge associated with the

US 7,711,857 B2

9

binding. The get key routine creates the key for the session associated with the message. If a key cannot be created until subsequent bindings are processed or because the current binding is to be removed, then the get key routine returns a next binding status, else it returns a continue status. In decision block **807**, if the return status of the get key routine is next binding, then the routine loops to block **802** to get the next binding, else the routine continues at block **808**. In block **808**, the routine invokes the routine get session. The routine get session returns the session associated with the key, creating a net session if necessary. In block **809**, the routine invokes the routine nail binding. The routine nail binding retrieves the binding if one is already nailed to the session. Otherwise, that routine nails the binding to the to session. In decision block **810**, if the nail binding routine returns a status of simplex, then the routine continues at block **811** because only one path can use the session, else the routine loops to block **802**. Immediately upon return from the nail binding routine, the routine may invoke a set map routine of the edge passing the session and a map to allow the edge to set its map. In block **811**, the routine invokes the find path routine, which finds the shortest path through the binding list and creates a path if necessary. In block **812**, the routine invokes the process path hopping routine, which determines whether the identified path is part of a different path. Path hopping occurs when, for example, IP fragments are built tip along separate paths, but once the fragments are built up they can be processed by the same subsequent path.

FIG. **9** is a flow diagram of the initialize demux routine. This routine is invoked to initialize the local data structures that are used in the demux process and to identify the initial binding. The demux routine finds the shortest path from the initial binding to the final binding. If the current status is demux extend, then the routine is to extend the path of the passed path entry by adding additional path entries. If the current status is demux end then the demux routine is ending the current path. If the current status is demux continue, then the demux routine is in the process of continuing to extend or in the process of starting a path identified by the passed address. In block **901**, the routine sets the local map structure to the map structure in the passed path entry structure. The map structure identifies the output label, the target label, and the target key. In the block **902**, the routine initializes the local message structure to the passed message structure and initializes the pointers path and address element to null. In block **903**, the routine sets of the variable saved status to 0 and the variable status to demux continue. The variable saved status is used to track the status of the demux process when backtracking to nail a binding whose nail was postponed. In decision block **904**, if the passed path entry is associated with a path, then the routine continues at block **905**, else the routine continues at block **906**. In block **905**, the routine sets the variable status to the status of that path. In block **906**, if the variable status is demux continue, then the routine continues at block **907**. If the variable status is demux end, then the routine continues at block **908**. If the variable status is demux extend, then the routine continues at block **909**. In block **907**, the status is demux continue, and the routine sets the local pointer path address to the passed address and continues at block **911**. In block **908**, the status is demux end, and the routine invokes the init end routine and continues at block **911**. In block **909**, the status is demux extend, and the routine sets the local path address to the address of the path that contains the passed path entry. In block **910**, the routine sets the address element and the current binding of the path address pointed to by the local pointer path address to the address entry of the member structure of the passed path entry. In the block **911**, the routine sets

10

the local variable status to demux continue and sets the local binding list structure to the binding list structure from the local path address structure. In block **912**, the routine sets the local pointer current binding to the address of the current binding pointed to by local pointer path address and sets the local variable postpone to 0. In block **913**, the routine sets the function traverse to the function that retrieves the next data in a list and sets the local pointer session to null. The routine then returns.

FIG. **10** is a flow diagram of the init end routine. If the path is simplex, then the routine creates a new path from where the other one ended, else the routine creates a copy of the path. In block **1001**, if the binding of the passed path entry is simplex (i.e., only one path can be bound to this bindings then the routine continues at block **1002**, else the routine continues at block **1003**. In block **1002**, the routine sets the local pointer path address to point to an address structure that is a copy of the address structure associated with the passed path entry structure with its current binding to the address entry associated with the passed path entry structure, and then returns. In block **1003**, the routine sets the local pointer path address to point to an address structure that on contains the URL of the path that contains the passed path entry. In block **1004**, the routine sets the local pointer element to null to initialize the selection of the bindings. In blocks **1005** through **1007**, the routine loops adding all the bindings for the address of the passed path entry that include and are before the passed path entry to the address pointed to by the local path address. In block **1005**, the routine retrieves the next binding from the binding list starting with the first. If there is no such binding, then the routine returns, else the routine continues at block **1006**. In block **1006**, the routine adds the binding to the binding list of the local path address structure and sets the current binding of the local variable path address. In the block **1007**, if the local pointer element is equal to the address entry of the passed path entry, then the routine returns, else the routine loops to block **1005** to select the next binding.

FIG. **11** is a flow diagram of a routine to get the next binding. This routine returns the next binding from the local binding list. If there is no next binding, then the routine invokes the routine label map get to identify the list of edges ("trails") that will map the output label to the target label. If only one trail is identified, then the binding list of path address is extended by the edges of the trail. If multiple trails are identified, then a path is created for each trail and the routine returns so that the demux process can be invoked for each created path. In block **1101**, the routine sets the local pointer binding to point to the next or previous (as indicated by the traverse function) binding in the local binding list. In block **1102**, if a binding was found, then the routine returns an indication that a binding was found, else the routine continues at block **1103**. In block **1103**, the routine invokes the label map get function passing the output label and target label of the local map structure. The label map get function returns a trail list. A trail is a list of edges from the output label to the target label. In decision block **1104**, if the size of the trail list is one, then the routine continues at block **1105**, else the routine continues at block **1112**. In blocks **1105-1111**, the routine extends the binding list by adding a binding data structure for each edge in the trail. The routine then sets the local binding to the last binding in the binding list. In block **1105**, the routine sets the local pointer current binding to point to the last binding in the local binding list. In block **1106**, the routine sets the local variable temp trail to the trail in the trail list. In block **1107**, the routine extends the binding list by temp trail by adding a binding for each edge in the trail. These bindings are not yet nailed. In block **1108**, the routine

US 7,711,857 B2

11

sets the local binding to point to the last binding in the local binding list. In decision block 1119, if the local binding, does not have a key for a session and the local map has a target key for a session, then the routine sets the key for the binding to the target key of the local map and continues at block 1110, else the routine loops to block 1101 to retrieve the next binding in path. In block 1110, the routine sets the key of the local binding to the target key of the local map. In block 1111, the routine sets the target key of the local map to null and then loop to block 1101 to return the next binding. In decision block 1112, if the local session is set, then the demultiplexing is already in progress and the routine returns a break status. In block 1113, the routine invokes a prepare multicast paths routine to prepare a path entry for each trail in the trail list. The routine then returns a multiple status.

FIG. 12 is a flow diagram of the get key routine. The get key routine invokes an edge's demux-key routine to retrieve a key for the session associated with the to message. The key identifies the session of a protocol. The demux key routine creates the appropriate key for the message. The demux key routine returns a status of remove postpone, or other. The status of remove indicates that the current binding should be removed from the path. The status of postpone indicates that the demux key routine cannot create the key because it needs information provided by subsequent protocols in the path. For example, a TCP session is defined by a combination of a remote and local port address and an IP address. Thus, the TCP protocol postpones the creating of a key until the IP protocol identifies the IP address. The get key routine returns a next binding status to continue at the next binding in the path. Otherwise, the routine returns a continue status. In block 1201, the routine sets the local edge to the edge of the local binding (current binding) and sets the local protocol to the protocol of the local edge. In block 1202, the routine invokes the demux key routine of the local edge passing the local message, local path address, and local map. The demux key routine sets the key in the local binding. In decision block 1203, if the demux key routine returns a status of remove, then the routine continues at block 1204. If the demux key routine returns a status of postpone, then the routine continues at block 1205, else the routine continues at block 1206. In block 1204, the routine sets the flag of the local binding to indicate that the binding is to be removed and continues at block 1206. In block 1205, the routine sets the variable traverse to the function to list the next data, increments the variable postpone, and then returns a next binding status. In blocks 1206-1214, the routine processes the postponing of the creating of a key. In blocks 1207-1210, if the creating of a key has been postponed, then the routine indicates to backtrack on the path, save the demux status, and set the demux status to demux continue. In blocks 1211-1213, if the creating of a key has not been postponed, then the routine indicates to continue forward in the path and to restore any saved demux status. The save demux status is the status associated by the binding where the backtrack started. In decision block 1206, if the variable postpone is set, then the routine continues at block 1207, else the routine continues at block 1211. In block 1207, the routine decrements the variable postpone and sets the variable traverse to the list previous data function. In decision block 1208, if the variable saved status is set, then the routine continues at block 1210, else the routine continues at block 1209. The variable saved status contains the status of the demux process when the demux process started to backtrack. In block 1209, the routine sets the variable saved status to the variable status. In block 1210, the routine sets the variable status to demux continue and continues at block 1214. In block 1211, the routine sets the variable traverse to the list

12

next data function. In decision block 1212, if the variable saved status is set, then the routine continues at block 1213, else the routine continues at block 1214. In block 1213, the routine sets the variable status to the variable saved status and sets the variable saved status to 0. In decision block 1214, if the local binding indicates that it is to be removed, then the routine returns a next binding status, else the routine returns a continue status.

FIG. 13 is a flow diagram of the get session routine. This routine retrieves the session data structure, creating a data structure session if necessary, for the key indicated by the binding. In block 1301, the routine retrieves the session from the session table of the local protocol indicated by the key of the local binding. Each protocol maintains a mapping from each key to the session associated with the key. In decision block 1302, if there is no session, then the routine continues at block 1303, else the routine returns. In block 1303, the routine creates a session for the local protocol. In block 1304, the routine initializes the key for the local session based on the key of the local binding. In block 1305, the routine puts the session into the session table of the local protocol. In block 1306, the routine invokes the create session function of the protocol to allow the protocol to initialize its context and then returns.

FIG. 14 is a flow diagram of the nail binding routine. This routine determines whether a binding is already associated with ("nailed to") the session. If so, the routine returns that binding. If not, the routine associates the binding with the session. The routine returns a status of simplex to indicate that only one path can extend through the nailed binding. In decision block 1401, if the binding table of the session contains an entry for the edge, then the routine continues at block 1402, else the routine continues at block 1405. In block 1409, the routine sets the binding, to the entry from the binding table of the local session for the edge. In block 1403, the routine sets the current binding to point to the binding from the session. In block 1404, if the binding is simplex, then the routine returns a simplex status, else the routine returns. Blocks 1405 through 1410 are performed when there is no binding in the session for the edge. In block 1405, the routine sets the session of the binding to the variable session. In block 1406, the routine sets the key of the binding to the key from the session. In block 1407, the routine sets the entry for the edge in the binding table of the local session to the binding. In block 1408, the routine invokes the create binding function of the edge of the binding passing the binding so the edge can initialize the binding. If that function returns a status of remove, the routine continues at block 1409. In block 1409, the routine sets the binding to be removed and then returns.

FIG. 15 is a flow diagram of the find path routine. The find path routine identifies the shortest path through the binding list. If no such path exists, then the routine extends a path to include the binding list. In decision block 1501, if the binding is simplex and a path already goes through this binding (returned as an entry), then the routine continues at block 1502, else the routine continues at block 1503, in block 1502, the routine sets the path to the path of the entry and returns. In block 1503, the routine initializes the pointers element and short entry to null. In block 1504, the routine sets the path to the path of the passed path entry. If the local path is not null and its status is demux extend, then the routine continues at block 1509, else the routine continues at block 1505. In blocks 1505-1508, the routine loops identifying the shortest path through the bindings in the binding list. The routine loops selecting each path through the binding. The selected path is eligible if it starts at the first binding in the binding list and the path ends at the binding. The routine loops setting the

US 7,711,857 B2

13

short entry to the shortest eligible path found so far. In block 1505, the routine sets the variable first binding to the first binding in the binding list of the path address. In block 1506, the routine selects the next path (entry) in the path list of the binding starting with the first. If a path is selected (indicating that there are more paths in the binding), then the routine continues at block 1507, else the routine continues at block 1509. In block 1507, the routine determines whether the selected path starts at the first binding in the binding list, whether the selected path ends at the last binding in the binding list and whether the number of path entries in the selected path is less than the number of path entries in the shortest path selected so far. If these conditions are all satisfied, then the routine continues at block 1508, else the routine loops to block 1506 to select the next path (entry). In block 1508, the routine sets the shortest path (short entry) to the selected path and loops to block 1506 to select the next path through the binding. In block 1509, the routine sets the selected path (entry) to the shortest path. In decision block 1510, if a path has been found, then the routine continues at block 1511, else the routine continues at block 1512. In block 1511, the routine sets the path to the path of the selected path entry and returns. Blocks 1512-1516 are performed when no paths have been found. In block 1512, the routine sets the path to the path of the passed path entry. If the passed path entry has a path and its status is demux extend, then the routine continues at block 1515, else the routine continues at block 1513. In block 1513, the routine creates a path for the path address. In block 1514, the routine sets the variable element to null and sets the path entry to the first element in the stack list of the path. In block 1515, the routine sets the variable element to be address entry of the member of the passed path entry and sets the path entry to the passed path entry. In block 1516, the routine invokes the extend path routine to extend the path and then returns. The extend path routine creates a path through the bindings of the binding list and sets the path status to the current demux status.

FIG. 16 is a flow diagram of the process of path hopping routine. Path hopping occurs when the path through the binding list is not the same path as that of the passed path entry. In decision block 1601, if the path of the passed path entry is set, then the routine continues at block 1602, else the routine continues at block 1609. In decision block 1602, if the path of the passed path entry is equal to the local path, then the routine continues at 1612, else path hopping is occurring and the routine continues at block 1603. In blocks 1603-1607, the routine loops positioning pointers at the first path entries of the paths that are not at the same binding. In block 1603, the routine sets the variable old stack to the stack list of the path of the passed path entry. In block 1604, the routine sets the variable new stack to the stack list of the local path. In block 1605, the routine sets the variable old element to the next element in the old stack. In block 1606, the routine sets the variable element to the next element in the new stack. In decision block 1607, the routine loops until the path entry that is not in the same binding is located. In decision block 1608, if the variable old entry is set, then the routine is not at the end of the hopped-from path and the routine continues at block 1609, else routine continues at block 1612. In block 1609, the routine sets the variable entry to the previous entry in the hopped-to path. In block 1610, the routine sets the path of the passed path entry to the local path. In block 1611, the routine sets the local entry to the first path entry of the stack list of the local path. In block 1612, the routine inserts an entry into return list and then returns.

Although the conversion system has been described in terms of various embodiments, the invention is not limited to

14

these embodiments. Modification within the spirit of the invention will be apparent to those skilled in the art. For example, a conversion routine may be used for routing a message and may perform no conversion of the message. Also, a reference to a single copy of the message can be passed to each conversion routine or demux key routine. These routines can advance the reference past the header information for the protocol so that the reference is positioned at the next header. After the demux process, the reference can be reset to point to the first header for processing by the conversion routines in sequence. The scope of the invention is defined by the claims that follow.

The invention claimed is:

1. A method in a computer system for processing packets of a message, the method comprising:

receiving a packet of the message and a data type of the message;

analyzing the data type of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence, wherein analyzing the data type of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received;

storing an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message;

for each of a plurality of components in the identified sequence:

performing the processing of each packet by the identified component; and

storing state information relating to the processing of the component with the packet for use when processing the next packet of the message.

2. The method of claim 1 wherein the receiving of the data type includes requesting the data type from a component that previously processed the packet.

3. The method of claim 1 wherein the component is a protocol with an edge.

4. A method in a computer system for processing a message, the message having a plurality of headers, the method comprising:

analyzing the plurality of headers of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence, wherein analyzing the plurality of headers of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received;

storing an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message;

for each of a plurality of components in the identified sequence:

performing the processing of each packet by the identified component; and

storing state information relating to the processing of the component with the packet for use when processing the next packet of the message.

US 7,711,857 B2

15

5. The method of claim 4 wherein the analyzing includes identifying a data type associated with each of the plurality of headers.

6. The method of claim 4 including locating state information based on information in each of the plurality of headers. 5

7. The method of claim 4, further comprising receiving from the identified component an identifier of state information associated with the each of the plurality of messages.

8. The method of claim 4 wherein analyzing the plurality of headers identifies a data type used to identify a component for processing a message. 10

9. The method of claim 4 wherein analyzing the plurality of headers identifies a data type used to identify the sequence of components for processing a message. 15

10. A computer readable storage medium, other than a data transmission medium, containing instructions for processing packets of a message, the instructions comprising at least one computer-executable module configured to:

20 receive a packet of the message and a data type of the message;

16

analyze the data type of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence, wherein analyzing the data type of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received;

store an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message;

for each of a plurality of components in the identified sequence:

perform the processing of each packet by the identified component; and

store state information relating to the processing of the component with the packet for use when processing the next packet of the message.

* * * * *

EXHIBIT 19

UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

---o0o---

IMPLICIT NETWORKS, INC.,

Plaintiff,

vs.

Case No.:
3:10-cv-4234 SI

JUNIPER NETWORKS, INC.,

Defendant.

/

VIDEOTAPED 30(b)(6) DEPOSITION OF

EDWARD BALASSANIAN

VOLUME 5

Thursday, August 16, 2012

REPORTED BY: RACHEL FERRIER, CSR 6948

(1-445620)

1 Okay. So this was previously marked
2 Exhibit 138.

3 MR. HOSIE: Thank you.

4 THE WITNESS: I can't write on these, can I?

5 MR. HOSIE: Actually --

10:32:27

6 THE WITNESS: Okay.

7 MR. McPHIE: You can. This is not the official
8 exhibit. It's a copy of one.

9 THE WITNESS: Okay.

10 MR. McPHIE: So if it's helpful, go for it.

10:32:31

11 Q If you look at page 11, this was in connection
12 with the selecting individual components limitation.

13 A So starting on page 10.

14 Q Starting on page 10 and then going on what I
15 would like to focus on -- and you are welcome to look at
16 the whole thing if you like. What I would like to focus
17 on is the very last sentence where the Court provides a
18 construction for selecting individual components.

10:33:04

19 A Okay. Let me just read this real quick.

20 Q Read the -- I mean, the whole thing is
21 obviously relevant, but let me know when you are
22 finished.

10:33:23

23 A Okay.

24 Q And let me pause there and go back a little
25 bit.

10:34:17

1 You will see, before the very end of the
2 paragraph, there is a statement which reads, in part,
3 beginning on line 7: A necessary part of selecting
4 individual components is determining the compatibility
5 between the output of one software routine and the input 10:34:34
6 of the next.

7 Do you see that there?

8 A Mm-hmm.

9 Q Do you agree with that statement?

10 MR. HOSIE: Objection; vague, ambiguous. 10:34:43

11 THE WITNESS: So there's a missing period in
12 this sentence. I'm assuming you are talking about
13 the -- the sentence up to the word "next"?

14 MR. McPHIE: Yes. I also assumed that a period
15 was intended there because of the capitalization of the 10:35:27
16 next word.

17 THE WITNESS: So we are only talking about the
18 first sentence there: The Court finds that, based on
19 the claim language teachings of the specification and
20 the prosecution history, a necessary part of selecting 10:35:36
21 individual components is determining compatibility
22 between the output of one software routine and the input
23 of the next, period, and you are asking me if I agree
24 with that.

25 BY MR. McPHIE: 10:35:49

1 Q Do you agree with that statement?

2 A I -- I do. Sure. Yes.

3 Q All right. Going to the next sentence, which
4 reads: Therefore, selecting individual components is
5 construed as, quote, selecting the individual software
6 routines of the sequence so that the input and output
7 formats of the software routines are compatible.

10:36:06

8 Do you see that there?

9 A I do.

10 Q And looking back the '163 patent, you will see
11 that claim 1 uses the words "such that" instead of "so
12 that."

10:36:21

13 Do you see that there?

14 A Mm-hmm.

15 Q In your mind, is there any significance or
16 distinction between the use of "such that" and "so
17 that"?

10:36:40

18 MR. HOSIE: Objection; calls for speculation,
19 asks for a legal opinion, asks that the witness construe
20 the Court's Construction Order.

10:36:50

21 THE WITNESS: I don't see a difference. I
22 don't see a difference between "so that" and "such
23 that." I think they --

24 MR. McPHIE: Okay.

25 THE WITNESS: -- imply the same thing.

10:37:04

1 BY MR. McPHIE:

2 Q You recall that we decided to refer to this
3 particular limitation from claim 1 of the '163 patent
4 that was identified earlier as the input/output matching
5 limitation; correct?

10:37:20

6 A Yes.

7 Q Do you understand the input/output matching
8 limitation to be substantially the same as what the
9 Court has set forth here regarding selecting individual
10 components?

10:37:40

11 MR. HOSIE: Same objections.

12 THE WITNESS: So are input/output matching
13 element is smaller than what this is, right? It doesn't
14 talk about selecting individual components.

15 MR. McPHIE: Actually, that's a good point, so
16 let me ask it again.

10:37:58

17 Q Part of the Court's construction of selecting
18 individual components is, quote, so that the input and
19 output formats of the software routines are compatible;
20 correct?

10:38:14

21 A Right. It says, Selecting individual software
22 routines of the sequence so that the input and output
23 formats of the software routines are compatible.

24 Q And I want to focus on the portion of that
25 construction that begins with "so that."

10:38:29

CERTIFICATE OF REPORTER

I, RACHEL FERRIER, a Certified Shorthand Reporter, hereby certify that the witness in the foregoing deposition was by me duly sworn to tell the truth, the whole truth, and nothing but the truth in the within-entitled cause;

That said deposition was taken down in shorthand by me, a disinterested person, at the time and place therein stated, and that the testimony was thereafter reduced to typewriting by computer under my direction and supervision and is a true record of the testimony given by the witness;

That before completion of the deposition, review of the transcript [X] was [] was not requested. If requested, any changes made by the deponent (and provided to the reporter) during the period allowed are appended hereto.

I further certify that I am not of counsel or attorney for either or any of the parties to the said deposition, nor in any way interested in the event of this cause, and that I am not related to any of the parties thereto.

DATED:

RACHEL FERRIER, CSR No. 6948

EXHIBIT 20

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Balassanian, et al.

Reexamination Control No.: 95/000,660

U.S. Patent No.: 7,711,857

Reexamination Request Filed: Mar. 2, 2012

For: METHOD AND SYSTEM FOR DATA
MULTIPLEXING

Examiner: Kenneth Whittington

Technology Center/Art Unit: 3992

RESPONSE

Attn: Mail Stop "Inter Partes Reexam"
Central Reexamination Unit
Office of Patent Legal Administration
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Office Action mailed May 10, 2012, please enter and consider the following Remarks showing that the three claims under reexamination are patentably distinguishable over the prior art references of record.

The pending claims are reflected in the Listing of Claims attached as an Appendix. No amendments have been made to the claims. Exhibits 1-2 are attached to this Response. Exhibit 1 is a declaration by Edward Balassanian, the inventor of the '857 patent. Exhibit 2 is a copy of the District Court's claim construction order in the related litigation.

The Patent Owner does not believe that extensions of time or other fees are required. However, if any fees are necessary to prevent abandonment of this reexamination, then such fees are hereby petitioned and hereby authorized to be charged to our Deposit Account No. 504592.

routers to accept it and continue the routing process. Only the content within the IP packet is encrypted, thus preserving the format of the packet as an IP packet. Although it is unclear from the Decasper paper, if the type of encryption is for a VPN link (which is only vaguely referenced in Decasper), then all encrypted entering packets would have to be decrypted before the Decasper system could classify them because the Decasper system requires a well-defined IP packet. Likewise, as explained above, if the encryption refers to an outgoing VPN connection, which is again unclear from the Decasper paper, this type of encryption would necessarily happen to packets leaving the Decasper system since no other plugins would be able to operate on the encrypted packets.

Third, as previously explained, there is no evidence (and the Requester has offered none) that the plugins in Decasper depend on any input or output constraints other than IP. In fact, it is clear that Decasper assumes and requires all incoming packets to be well-formatted IP packets for the packet classification system to work properly. Even the example of encryption does not follow because the gates do not know if the previous plugin did or did not encrypt the packet—they merely assume IP packets. Thus, there would be no motivation to add the missing limitation.

More particularly, there would have been no “reason”—as required by *KSR*—to consider format compatibility in Decasper. The architecture is specifically designed to handle one format (IP) and, thus, one of ordinary skill would be unconcerned about matching different formats. In fact, adding functionality to check or compare the output format of a packet processed by one IP component with the input format required by the next IP component, would add unnecessary overhead. In short, no skilled person looking at Decasper would add the claim requirement.

Furthermore, not only is there no reason to add format compatibility to Decasper, the Examiner fails to cite any evidence for adding such a feature. There is no cited patent or printed publication or any other document; instead, there are only bald assertions without the evidentiary support required by the law. This is insufficient under the law and the MPEP as discussed in Section V(B) above. For at least the foregoing reasons, Decasper fails to render obvious claims 1, 4 and 10. In addition, even if, *arguendo*, one of ordinary skill would be inclined to modify Decasper to include the disputed limitation, the fact remains that Decasper as modified still fails to teach or suggest multiple other limitations of claims 1, 4 and 10, as Patent Owner explains above under Issue 1. Patent Owner therefore respectfully requests that the Examiner withdraw the obviousness rejection under Issue 2.

VII. ISSUE 7: The Patentability of All Claims Should Be Confirmed Over the Combination of Decasper and IBM

The Examiner states that his rejection under Issue 7 is a combination of Decasper and IBM. This statement conflicts with the fact that the Examiner actually relies on Nelson (on p. 12 of the Office Action) in addition to Decasper and IBM. Thus, it is unclear what the Examiner's precise grounds for rejection are under Issue 7. For purposes of responding to Issue 7, Patent Owner addresses only Decasper and IBM, since Nelson does not appear to be part of the Examiner's official ground of rejection under Issue 7. In fact, given that the Examiner makes an obviousness rejection using Decasper, IBM, and Nelson under Issue 8, Patent Owner respectfully requests that the Examiner withdraw the rejections under Issue 7 altogether.

The rejections under Issue 7 should also be withdrawn because the combination of Decasper and IBM fails to render obvious the claims. Specifically, claim 1 requires, "for each of a plurality of components in the identified sequence," "storing state information relating to the processing of the component with the packet for use when processing the next packet of the

on IP protocol, would have to perform its compression and decompression “stateless[ly]” and on each individual datagram with absolutely no regard for the compression or decompression of any other datagram. This flies directly in the face of the claims, which require “storing state information relating to the processing of the component with the packet for use when processing the next packet of the message.” For at least this reason, one of ordinary skill in the art would not have been motivated, and indeed would have been unable, to implement any kind of “stateful” compression at all in Decasper so as to render the claims obvious.

Second, the compression and decompression techniques taught in IBM (*e.g.*, LZ-based schemes, such as LZ77) are dictionary-based techniques that are incompatible with Decasper. That is, they require the use of a common dictionary at the compression end and the decompression end to ensure that compressed data is properly decompressed. The Examiner acknowledges the use of such a dictionary in the Office Action at p. 12. Without the benefit of such a dictionary, the decompression end would not know how to decompress the compressed data. Such dictionaries must be transmitted from the compression end to the decompression end before the compressed data is sent. But such transmission of dictionaries requires link layer protocols such as the Point-to-Point protocol (PPP) and, indeed, the only context in which IBM describes LZ77 is the PPP. PPP is used to connect point-to-point links and requires connections such as ISDN, dial-up, etc. However, one of ordinary skill would have recognized that such connections are wholly incompatible with Decasper’s gates, since Decasper’s gates use only Internet Protocol (IP). Thus, Decasper’s routing technology is incompatible with LZ technologies like the LZ77 technique that IBM describes. Decasper and IBM cannot be combined for this additional reason. And even if Decasper routers were to use LZ77, the only way this could possibly work would be for outgoing data (*i.e.*, data exiting the Decasper system)

to be compressed such that no other subsequent gates in the Decasper router would process the compressed packets. Similarly, if the Decasper router received data compressed with LZ77, it would have to be decompressed before the Decasper system could classify the packet since the Decasper system assumes and requires well-formatted IP packets.

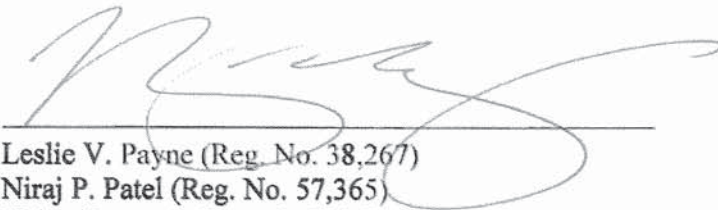
Further, even if Decasper and IBM could be combined (which they cannot), the resulting combination still would not teach or suggest the claimed state information limitation. The claims do not merely require the storage of state information to process data, as the Examiner suggests. Instead, the claims require the storage of state information such that state information relating to the processing of one packet can be used to process a subsequent packet. The combination of Decasper and IBM does not teach or suggest this kind of state information. In fact, even if IBM's dictionaries/tables could be used in Decasper (which, for the reasons stated above, they cannot), such dictionaries/tables would only comprise "state" information that is used in the same way to decompress each received packet regardless of which message contains the packet, with all packets being decompressed independently of each other. In other words, it is global state information applied to all packets of all messages, not packetized state information that is specific to a given message, as the claims require.

Based on the foregoing, the (untenable) combination of Decasper and IBM fails to render obvious claims 1, 4 and 10. In addition, even if, *arguendo*, one of ordinary skill were inclined to modify Decasper with IBM to include the disputed limitation, the fact remains that the combination of Decasper and IBM results in a system that lacks multiple other limitations of claims 1, 4 and 10, as Patent Owner explains above with respect to Issue 1. The Examiner should thus withdraw the obviousness rejections against claims 1, 4 and 10 in view of Decasper and IBM.

XI. CONCLUSION

For the reasons set forth above, Patent Owner Implicit respectfully requests the PTO to withdraw its rejections of claims 1, 4, and 10 and to confirm the patentability of all claims.

Respectfully submitted,

A handwritten signature in dark ink, appearing to read 'Leslie V. Payne', is written over a horizontal line.

Leslie V. Payne (Reg. No. 38,267)

Niraj P. Patel (Reg. No. 57,365)

Heim, Payne & Chorush, L.L.P.

600 Travis Street, Suite 6710

Houston, Texas 77002

Phone: (713) 221-2000

Fax: (713) 221-2021

Dated: July 10, 2012